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

YURIKO HASHIMOTO

990853225

PENN STATE

MGIS - CAPSTONE

AUGUST 5, 2019

CONTENTS

ACKNOWLEDGEMENTS

Throughout this project I have been provided with a great deal of support and assistance. Foremost, I sincerely thank my advisor Dr. Julian Avery, Beth King, and Dr. Justine Blanford at Penn State for their invaluable guidance and support. I express gratitude to the Canadian Wildlife Service and Fisheries and Oceans Canada who as my employers provided the time and resources to carry out and complete the study. My CWS colleagues—Andre Breault, Kathleen Moore, Dr. Andrea Norris, Jeffrey Thomas and Dr. Caroline Fox—provided indispensable guidance and encouragement and were ever generous with their time and wealth of knowledge. Finally, I thank my partner Brady Ciel Marks for her curiosity, intelligence, and willingness to come along on the deep dives into R.

ABSTRACT

Conservation biology has seen an explosion of species distribution models (SDMs) in the recently 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 population 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 map predictive distributions for ten species and five species groups within each of the eight ecosections of the central interior representing core provincial breeding waterfowl habitat. The developed methods, techniques, tools, and recommendations form a template for future research and act as guides in species distribution modeling.

Keywords: waterfowl, random forest, species distribution model, population monitoring

INTRODUCTION

Species distribution models (SDM) are GIS-based tools to predict where species are likely and unlikely to occur. Understanding species distributions is fundamental to effective conservation strategies (Guillera-Arroita et al. 2015; Guisan et al. 2013; Guisan and Thuiller 2005). By incorporating species life-history, observational, and environmental data, SDMs can provide ecological insights into habitat preferences and predict distributions across landscapes of space and time. Mapping the species-habitat relationship can facilitate targeted conservation action in response to conservation challenges such as climate change adaptation, 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 to help address the urgent conservation issues 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. Barker et al's models performed well overall however, as the authors cautioned, the results of extrapolation to out-of-sample areas were variable due to the inherent weakness of machine-learning methods to extrapolate to conditions for unknown environmental conditions outside of those sampled (Barker et al, 2014). British Columbia is not within the traditional survey area hence no survey data from BC informed their models which predicted low abundances and smoothed patterns of distribution for BC that conflicted with anecdotal expert opinion. Additionally, BC's geography is spectacularly diverse and distinct from the rest of Canada. 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 standardized waterfowl breeding population distribution models in common use for the province.

In the early 2000s after exploratory pilot surveys in BC's central and sub-boreal plateaus 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, assess the annual population status of migratory game birds in the central interior and contribute to adaptive harvest

strategies for mallards in the Pacific Flyway (Zimpfer, Breault, and Sanders 2019). Together, the annual surveys form a long-term monitoring dataset with the potential to inform more than the development of annual hunting regulations.

The aim of this study was to realize the collective value of the BC May surveys and create SDMs in support of the Canadian Wildlife Service's (CWS) mandate to conserve biodiversity (Environment and Climate Change Canada 2019; Environment Canada - Biodiversity Convention Office 1995). I developed random forest-based SDMs for the top ten most abundant waterfowl species as well as five group classifications based on community descriptors for BC's central interior (Table 1). Guisan et al (2013) have observed that the wide-spread development of SDMs has come with little guidance on their application. I follow their recommendation that SDMs be constructed within a practice-oriented framework, and provide guidance on how the models may be developed, improved upon, and implemented. In doing so, I highlight the power and utility of GIS as a decision-making platform for conservation planning by the CWS and partners in conservation.

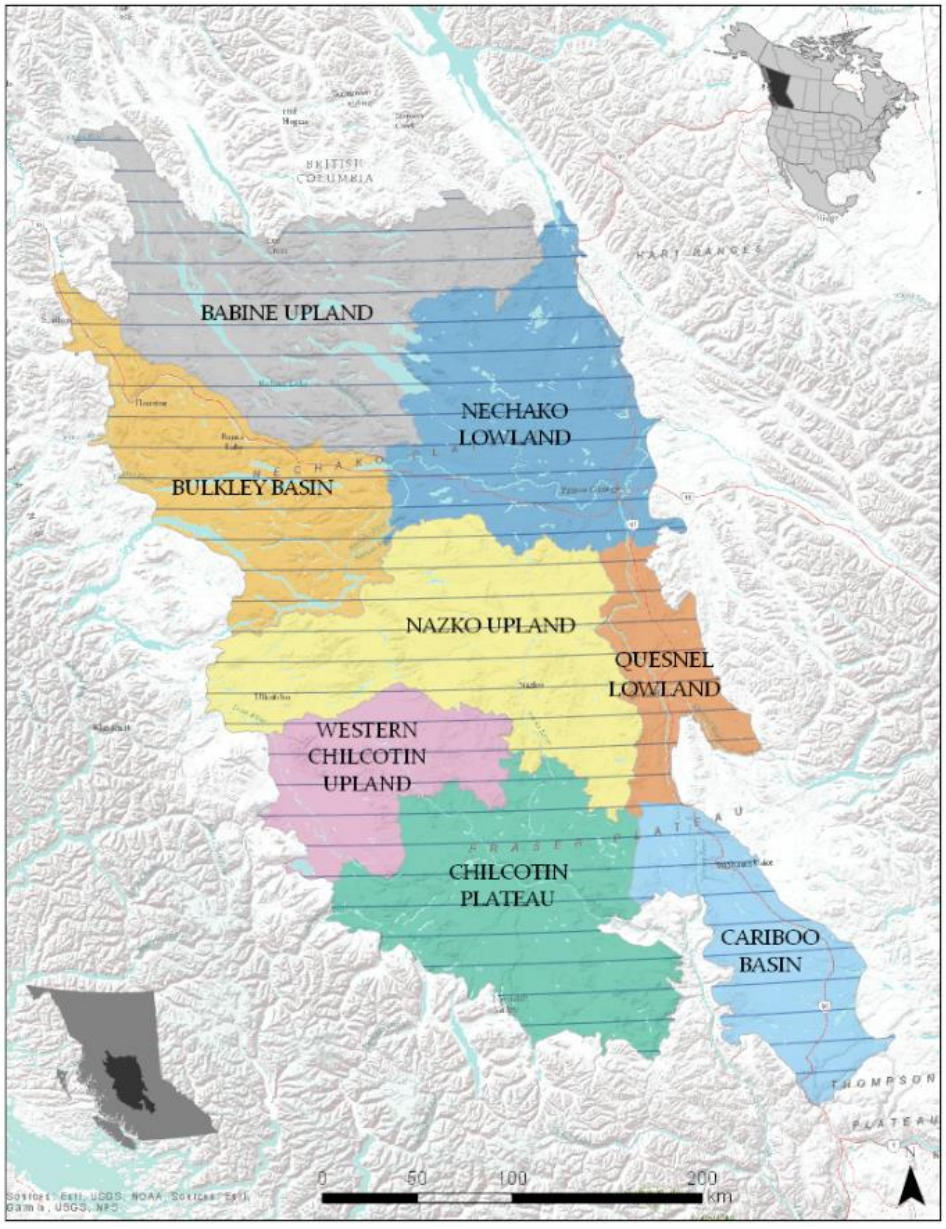


Figure 1. The survey area divided into the ecosections of the Ecoregions of British Columbia Classification system (Demarchi, 2011). Latitudinal strip transects (horizontal lines) are spaced ten miles (~16 km) apart within the Central and Sub-boreal plateaus of BC.

MATERIALS AND METHODS

Survey Methods

The BC Waterfowl Survey (also referred to as the "May surveys") was designed to capture the waterfowl breeding habitat of the humid, continental plateaus of the central and sub-boreal interior between the Coast Mountains to the west and the Rocky Mountains to the east. The region of interlocking highlands and valleys are mainly forested and sparsely populated. The main industries include forestry, ranching, mining, and cereal-crop agriculture in low-lying valleys (Demarchi 2011).

The survey design consists of latitudinal strip transects 400 metres wide, spaced ten miles apart (16.09 km), delineated by the boundaries of eight ecosections which represent the finest scale, sub-regional stratum of 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). 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 a region of flat and rolling hills that contain numerous meandering streams and low-lying depressions which create an abundance of wetland habitat (Demarchi 2011). 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.

Survey start dates are designed to capture the primary breeding period beginning in early May, but are weather and climate-dependent as heavy snowpack and cold spring temperatures can delay accessibility to open water and wetlands. Survey periods have started as early as late April (April 28, 2015) and as late as mid-May (May 13, 2011). The study area extent is almost 11 million hectares of which ~ 2.7% is directly surveyed taking an average 23 days (or ~105 flying hours) to complete.

The survey techniques employed are consistent with the methods of the traditional WBPHS outlined in Smith (1995) and Standard Operating Procedures (US Fish and Wildlife Service (USFWS) and Canadian Wildlife Service (CWS 1987). Prior to 2010, the survey technologies 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 has replaced paper analog methods.

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

Ecoregion	Ecosection	Topography	Climate/Vegetation	Indigenous Governance, Protection, Development
Fraser Basin	 Babine Upland BAU	Rolling upland with low ridges; many small streams and lakes and several large lakes Total Area 19, 620 km^2	Sub-continental--extreme winter cold and snow events, humid and rainy Sub-boreal spruce dominated forests	Carrier Sekani Tribal Council (Nadleh Whuten, Nak'azdli Whut'en, Tl'azt'en Nation), Lake Babine Nation, McLeod Lake Indian Band, Tsay Keh Dene Band, Wet'suwet'en Nation, Yekooche First Nation Sparsely populated, seasonal lodges; many protected areas
Fraser Plateau	 Bulkley Basin BUB	Broad, lowland valleys with many lakes Total Area 13, 360 km^2	Rainshadow effect of Coast mountains Dominated by lodgepole pine; trembling aspen in lower south-facing valleys	Carrier Sekani Tribal Council, Ts'il Kaz Koh First Nation (Burns Lake Band), Nadleh Whuten First Nation, Stellat'en First Nation, Wet'swewet'en First Nation, Cheslatta Carrier Nation, Lake Babine Nation, Wet'suwet'en Nation, Nee-Tahi-Buhn Band, Skin Tyee Nation, Witset First Nation Extensive development and agriculture within Yellowhead Hwy corridor and surrounding Francois Lake; protected areas include Francois lake Park
Fraser Plateau	 Cariboo Basin CAB	Rolling plateau with gentle sloping ; many streams, wetlands and lakes Total Area 9,035 km^2	Warmest temperatures within Fraser Plateau; relatively high summer precipitation Dry forests with many wetlands and grasslands on south-facing slopes; mainly Douglas-fir forests with lodgepole pine at higher elevations	?Esdilagh First Nation, Bonaparte, Pellt'iq't First Nation (Whispering Pines/Clinton), Esk'etemc First Nation, Northern Shuswap Tribal Council, Canim Lake Band, Soda Creek Indian Band, Stswecem'c Xgat'tem First Nation, T'exelcemc (Williams Lake Indian Band) 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 Total Area 16,600 km^2	Pronounced rain shadow effect rivers and lodgepole pine at higher elevations	Da'naxda'xw Awaetlatla Nation, Esk'etemc First Nation, Homalco Indian Band, Northern Shuswap Tribal Council, Alexis Creek First Nation, Tl'etinqox Government, Yenesit'in Government Ranching and logging are main industries; Big Creek Park is largest park
Fraser Plateau	 Nazko Upland NAU	Rolling uplands with many areas of high relief; slow moving streams and wetlands Total Area 18,150 km^2	Sub-continental climate--cold winters, warm summers, maximum precipitation in late spring/early summer; subject to cold blasts of Arctic air Lodgepole pine in south and white spruce/lodgepole pine/subalpine fir in north	Carrier Sekani Tribal Council, Cheslatta Carrier Nation, Lheidli T'enneh First Nation, Nazko First Nation, Lhoosk'uz Dene Nation, Ulkatcho First Nation, Saik'uz First Nation Ranching and First Nations reserves but no large communities

Fraser Basin		Nechako Lowland NEL	Flat or gently rolling lowlands with many small lakes, streams and wetlands Total Area 16,780 km^2	Sub-boreal climate- humid summers and harsh winters Sub-boreal spruce forests and lodgepole pine	Carrier Sekani Tribal Council, Lheidli T'enneh First Nation, McLeod Lake Indian Band, Nazko First Nation, Yekooche First Nation, Lhoosk'uz Dene Nation, Ulkatcho, Saik'uz First Nation, Nazko First Nation Largest city is Prince George; cereal crop farming in southern low-lying areas and ranching
Fraser Plateau		Quesnel Lowland QUL	Lowland trench bissected by the Fraser River; small lakes and wetlands but no large lakes Total Area 5,310 km^2	Subject to cold Arctic air; increased precipitation as air moves over Columbia Mtns and in summer due to upland wetlands Douglas-fir on dry south-facing slopes and trembling aspen, lodegpole pine, white spruce to subalpine fir with increasing elevation	?Esdilagh First Nation, Lhtako Dene Nation, Nazko First Nation, Lheidli T'enneh First Nation, Northern Shuswap Tribal Council Quesnel is the largest and only city but many smaller communities; farming, ranching and forestry (with extensive logging)
Fraser Plateau		Western Chilcotin Upland WCU	Rounded uplanded with a large lowland area containing many lakes Total Area 8,290 km^2	Sub-continental climate subject to blasts of Arctic winter air Mainly lodgepole pine and white spruce in higher elevations	Nazko First Nation, Ulkatcho First Nation Anahim Lake is largest community

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 group.

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

Species Code	Common Name	Scientific Name	Species Group	Foraging Guild	Nesting Guild
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	Description
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

Surveys are conducted by helicopter along meandering paths within the strip transect. The helicopter is flown at altitudes and speeds lower than fixed-wing aircraft (30-50m and 40-80 km/h, respectively) and bird detection is assumed complete. Therefore no complementary ground surveys are conducted and no visibility correction factor is applied. The total indicated breeding population estimates were derived from raw counts based on species, sex, and social group 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. To account for this spatial uncertainty, observation location points were collated to the nearest 400m interval along the center line of the strip transect.

The greatest total abundance and density of waterfowl occur in the Cariboo Basin ecosection which contains the productive wetlands of the Riske Creek area (Demarchi 2011; Savard, Sean Boyd, and John Smith 1994) (Figure 2). After Cariboo Basin the most densely populated ecosections include the Chilcotin Plateau, Nechako Lowland, and Nazko Upland in decreasing order (Figure 3). The remaining ecosections—Bulkley Basin, Quesnel Lowland, Western Chilcotin Upland, and Babine Upland—have similar population densities with Babine Upland consistently the least populous. These latter 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. A closer inspection of species-specific trends indicate these population fluctuations may be due mainly by mallard, scaup, ring-necked duck and green-winged teal

populations (Figure 4) with the apparent increasing trend driven by scaup, ring-necked duck, bufflhead and American wigeon (Figure 5).

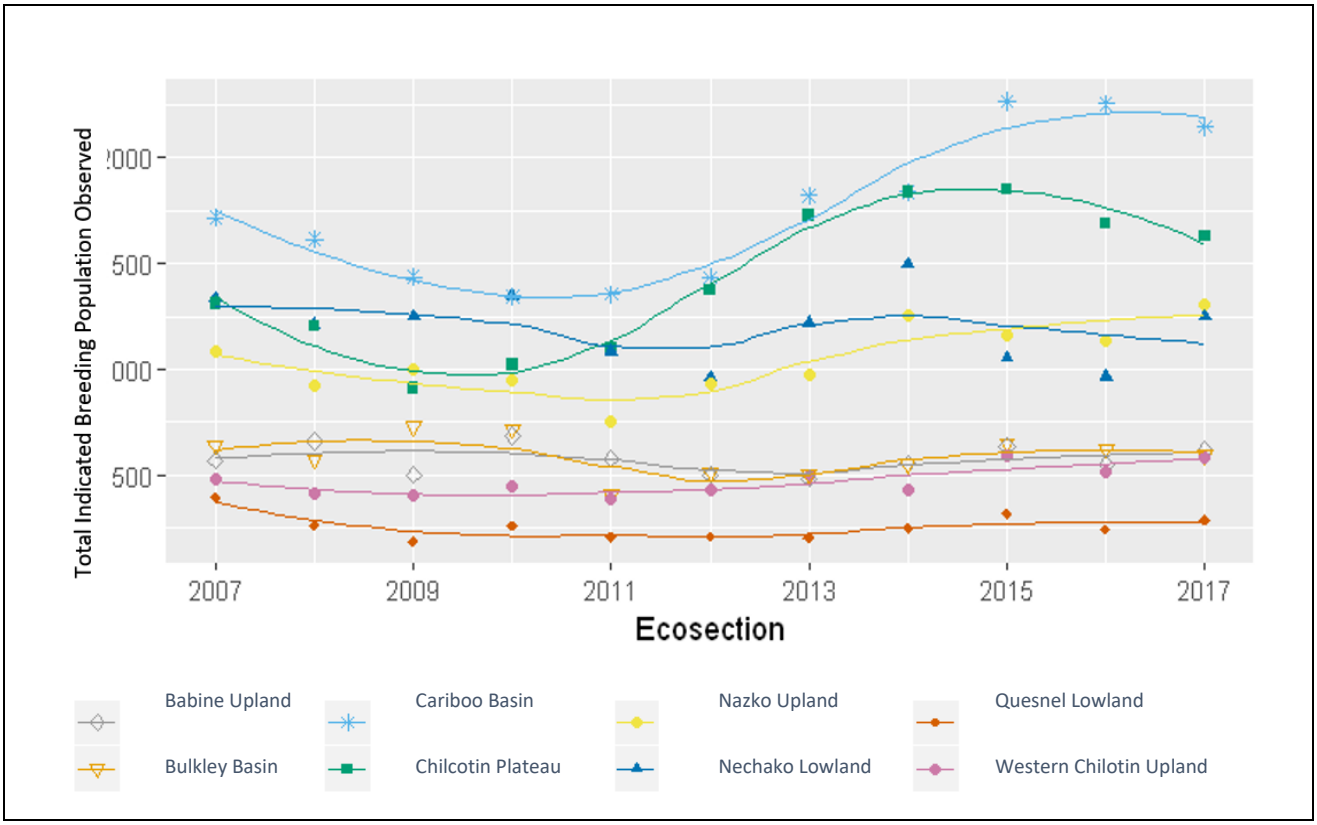


Figure 2. Combined total indicated breeding population of the top ten most common species observed within the transect (2007-2017).

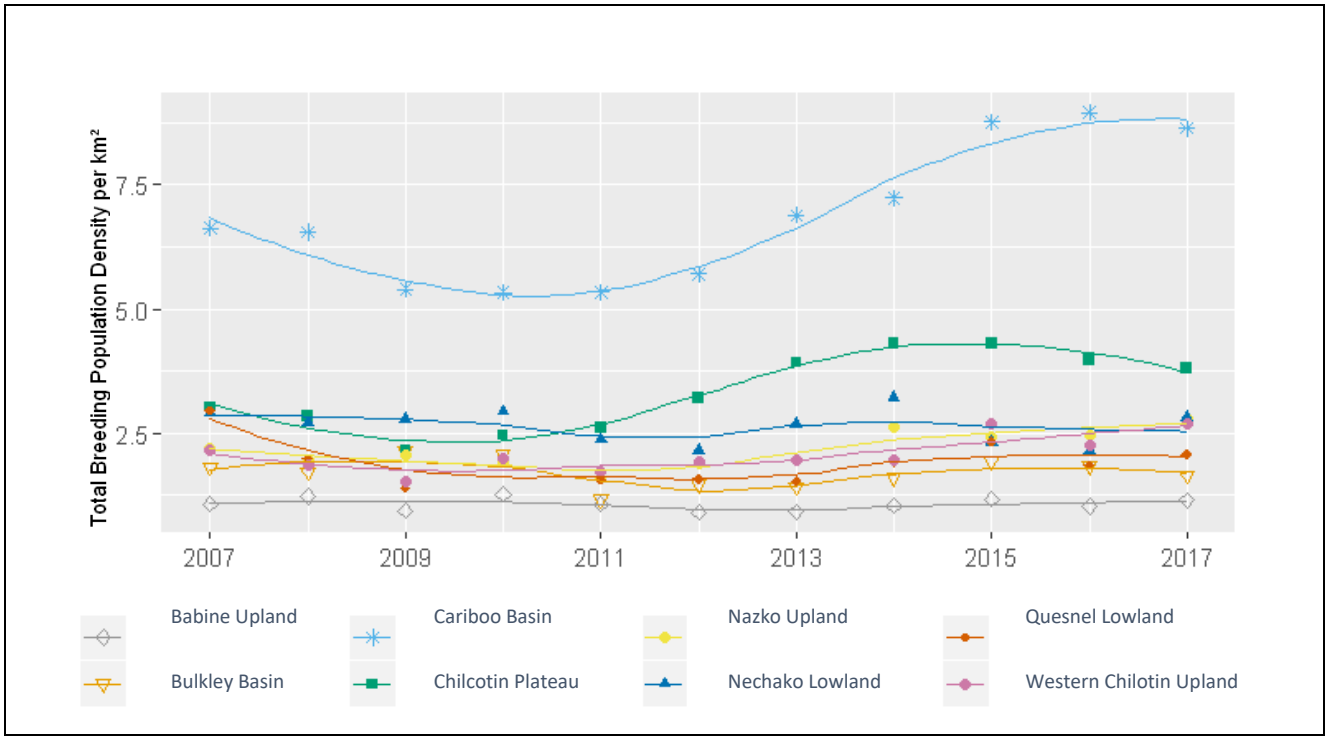


Figure 3. Combined total annual indicated breeding population density per square kilometre of the top ten most common species by ecosection observed within the transect.

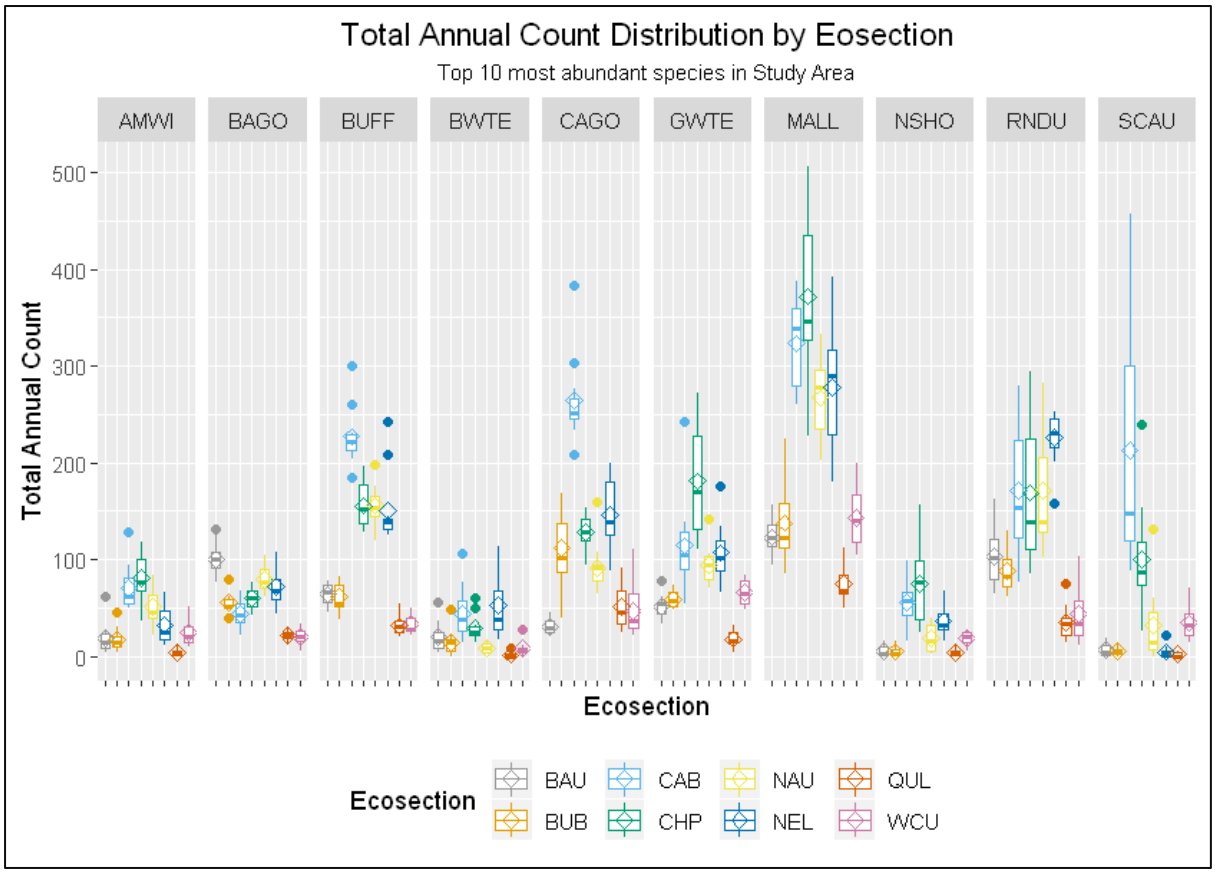


Figure 4. Boxplots of total annual indicated breeding population of the top ten most common species by ecosection observed within the survey transects.

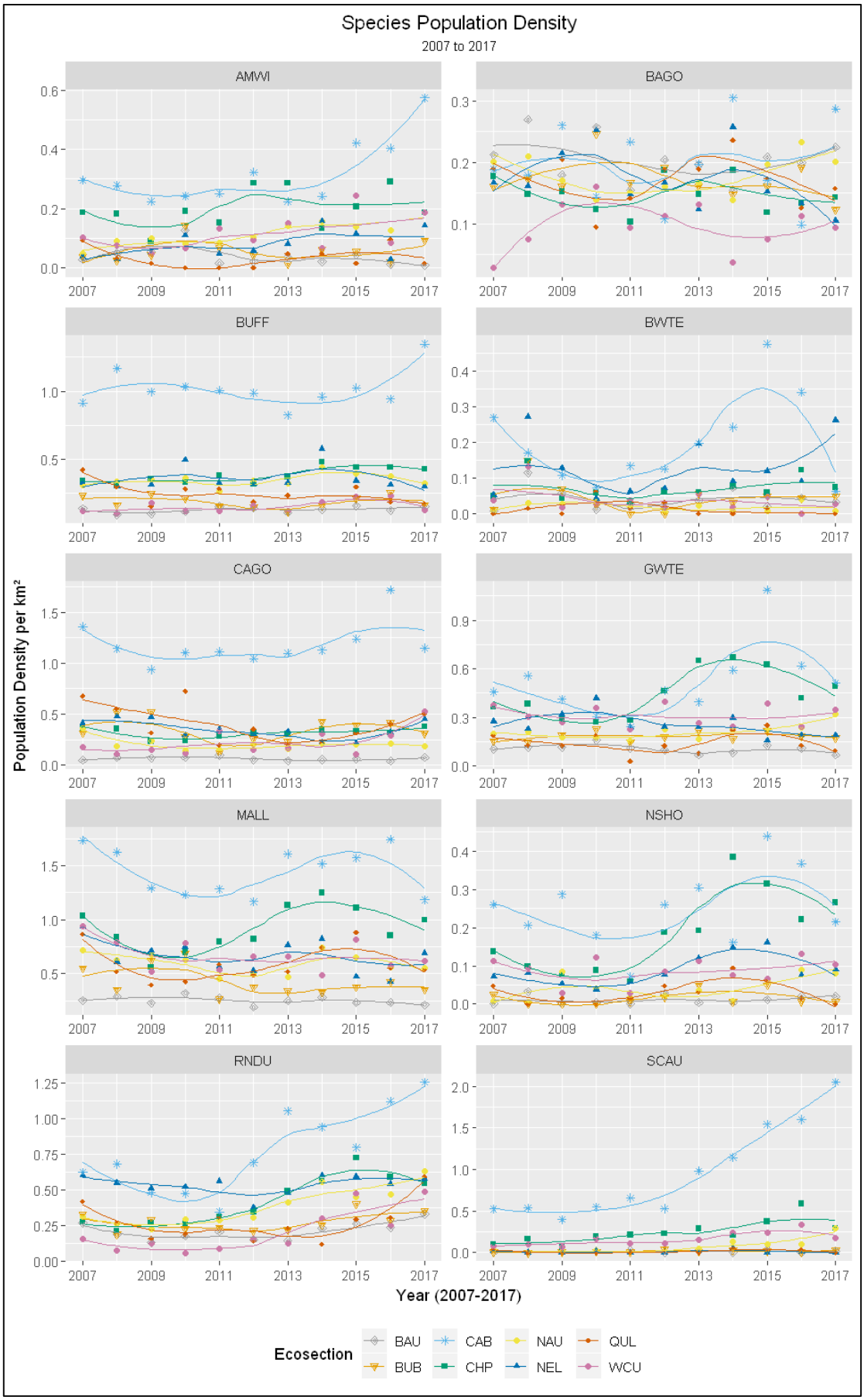


Figure 5. Population trends of the ten most abundant species over the survey period.

Environmental data

Environmental predictor variables were pre-selected based on ecological theory of life history traits and physiological processes(Table 2). Where possible direct measures of predictor variables rather than proxy data were selected. For example, measures of temperature and precipitation rather than indirect measures of elevation were included in the analyses. Several 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 such 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 centerline. 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 accumulation 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 vegetation (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 System (CCRS/CCMEO/NRCan 2017). Cover classes were aggregated and reclassified to account for imbalanced classes (Appendix 2). Topographic data were derived from 1:20,000 DEM (FLNRO, 2014) and included slope which indicates potential for water accumulation and aspect which influences solar radiation and temperature and can influence vegetation and soil development.

Hydrological variables associated with wetlands were derived from the Freshwater Atlas, 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 3). 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)—a provincial land use zone 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.

Category	Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation
Hydrology	fwa_1	Freshwater Atlas - Lakes < 1 ha	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (Variable)	Data BC
	fwa_2	Freshwater Atlas - Lakes 1-2 ha	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (variable)	Data BC
	fwa_3	Freshwater Atlas - Lakes 3-5 ha	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (variable)	Data BC
	fwa_4	Freshwater Atlas - Lakes 5-10 ha	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (variable)	Data BC
	fwa_w	Freshwater Atlas - Wetlands	Total area of wetlands within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (variable)	Data BC
	fwa_r	Freshwater Atlas - River	Total area of rivers within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (variable)	Data BC
	fwa_10ha_plus	Freshwater Atlas - Lakes > 10 ha	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (variable)	Data BC
	shorecx_10less	Freshwater Atlas - Shoreline index of lakes < 10 ha	Ratio of total perimeter to area of lakes, multiplied by 100,000	1:20,000	Static (variable)	Data BC
	shorecx_10plus	Freshwater Atlas - Shoreline index of lakes > 10 ha	Ratio of total perimeter to area of lakes, multiplied by 100,000	1:20,000	Static (variable)	Data BC
	str_s	Freshwater Atlas - Stream order 1-3	Total length of streams of stream orders 1 to 3	1:20,000	Static (variable)	Data BC
	str_m	Freshwater Atlas - Stream orders 4-6	Total length of streams of stream orders 4 to 6	1:20,000	Static (variable)	Data BC
Topography	aspect	Compass direction - indirect measure of productivity	Average aspect within 400 x 400 m areal interval of strip transect	1:20,000	Static (2011)	Data BC
	slope	Slope – potential wetland formation	Average slope within 400 x 400 m areal interval of strip transect	1:20,000	Static (2011)	Data BC
Land Management	alr	Agricultural Land Reserve – cropland	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (2014)	Data BC
	pa	Protected Areas – aggregated	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (2018)	CWS Conservation Areas DB
Land Cover	dra_u	Digital Road Atlas, Unpaved Roads	Total density within 1.2 km circular radius of centroid point along 400m interval of transect	1:20,000	Static (2014)	Data BC
	urban	Land cover - urban	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	30 m	Static (2010)	Commission for Environmental Cooperation

Category	Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation
	grassland	Land cover – grassland	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	30 m	Static (2010)	Commission for Environmental Cooperation
	shrubland	Land cover - shrubland	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	30 m	Static (2010)	Commission for Environmental Cooperation
	mixed_forest	Land cover - mixed forest	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	30 m	Static (2010)	Commission for Environmental Cooperation
	broadleaf	Land cover - broadleaf (deciduous)	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	30 m	Static (2010)	Commission for Environmental Cooperation
	needleleaf	Land cover - needleleaf (coniferous)	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	30 m	Static (2010)	Commission for Environmental Cooperation
Bioregional Classification	bec_zn	Biogeoclimatic Zone	Majority area within 400 x 400 m areal interval of strip transect	1:20,000	Static (2018)	Data BC
	eco	Ecosection – model extent	Rasterized within polygon boundaries	1:20,000	Static (2006)	Data BC
Climate	norm_dd5	30 yr normal Growing Degree Days > 5°C	Climatic measure at coordinate position and elevation along 400m interval of transect	Scale-free 400 m	1980-2010	ClimateBC
	norm_cmd	30 yr normal Climatic Moisture Deficit – indirect measure of soil moisture	Climatic measure at coordinate position and elevation along 400m interval of transect	Scale-free 400 m	1980-2010	ClimateBC
	pas_wt	Mean Precipitation as Snow in winter – influences catchment hydrology and stream flow	Climatic measure at coordinate position and elevation along 400m interval of transect averaged over survey period	Scale-free 400 m	Annual	ClimateBC
	tave04	Average April Temperature – influences snow melt catchment hydrology and stream flow	Climatic measure at coordinate position and elevation along 400m interval of transect averaged over survey period	Scale-free 400 m	Annual	ClimateBC

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).

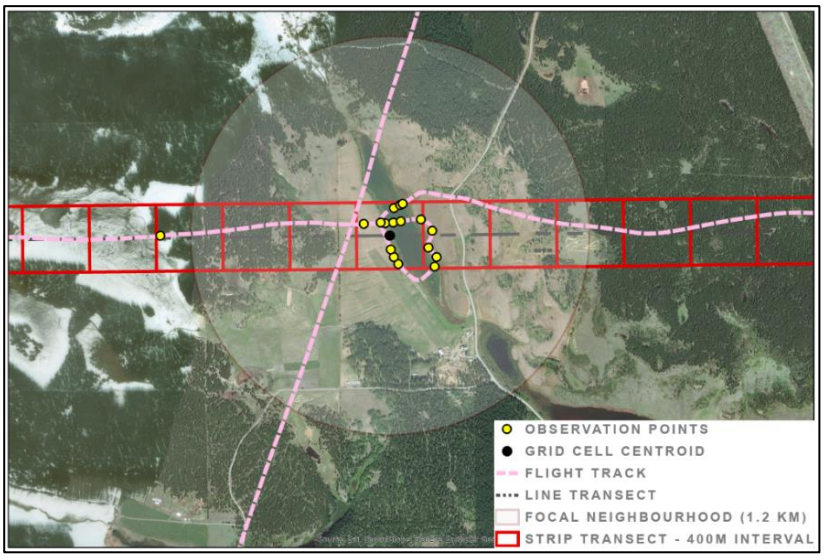


Figure 6. Representative schematic of survey methods incorporating data from 2017. Observation points (yellow dots) are collected along a meandering flight tracklog (dashed pink line).The strip transect (horizontal red line) was spliced into 400m wide interval.

Statistical Methods

Random forest is an ensemble machine learning algorithm that creates a series of decision trees based on the principles of random permutation and bagging[1]. Each individual decision tree is based on a random subset of the data. 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). However, by combining and averaging the results of a mulititude of 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 were 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

[1] Bagging is also known as bootstrapped aggregation which refers to the drawing of a large number of data samples by random sampling with replacement

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 that 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 and species group were developed for each ecosection (8 ecosections, 15 species/species groups for 120 models). Predicted abundance and distribution for each model were mapped to the full study area extent based on the environmental values sampled for each cell. Preliminary trials of models run on the full study area extent indicated ecosection was a significant environmental predictor warranting an ecosection-based design. Additionally, the computational intensity required for the full extent greatly exceeded the available resources and subsetting the data to the ecosection level facilitated ease of analyses.

The conditional inference trees utilized by 'party'package are insensitive to highly correlated data structures but in order to reduce processing time and facilitate interpretation, data preparation involved 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 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 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 5.

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. Predictor variables were ranked high to low (1-27) by their relative value within each species-ecosection model.

The top-ranked predictors averaged over all models were, as expected, hydrological (Figure 7). In order of decreasing importance were the direct measures of shoreline to lake area index of lakes less than 10 hectares in size (shorecx_10less), area of wetland (fwa_w), shoreline index of lakes greater than 10 hectares (shorecx_10plus), and the indirect measures of wetland availability, the climatic moisture deficit (cmd) and slope which is indicative of potential water pooling. Small streams (aggregated segments of streams orders 1-3), primary productivity indicated by growing degree days over 5 degrees celsius, needleleaf land cover, sub-hectare lakes, and lakes 3-5 ha in size rounded out the top ten (see Table 5 for a summary of ranks by ecosection). Lake size classes smaller than 10 ha do not appear in the overall top five 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 hectares indicated a significant influence that warranted their segmentation into smaller classes. Predictors with the lowest average importance ranking in increasing order included protected areas (pa), rivers, BEC zone, Agricultural Land Reserve (alr), urban and grassland land cover—each of which have low or near-zero variance within some ecosections and/or are patchily distributed.

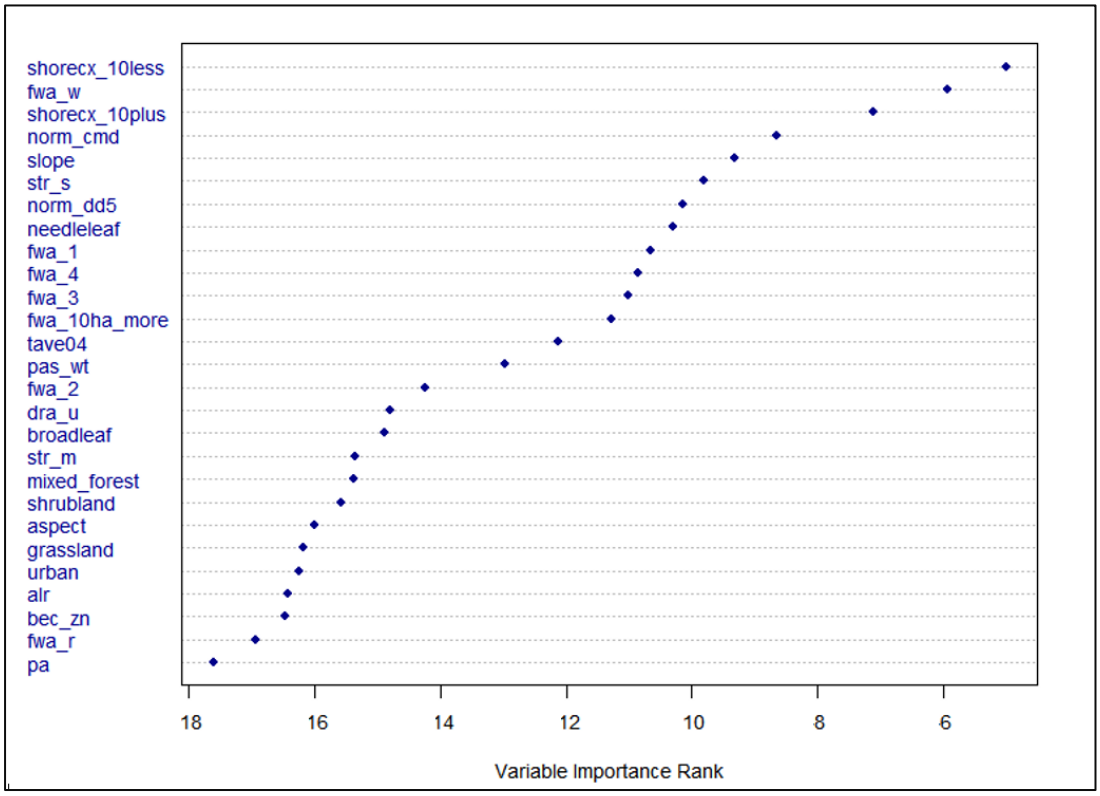


Figure 7. Variable importance rank of predictor variables averaged over all of the species-ecosection models.

The average relative importance rank of predictors was variable between ecosections (Figure 8), and between species-ecosection models (Figures 9 A-H). For example, lakes in size class greater than 10 hectares, urban land cover, unpaved roads, BEC zone and rivers, especially, were outliers that ranked highly in Western Chilcotin

Upland but poorly elsewhere. The range in relative rank between ecosections were smaller for habitat-specialist species such as Barrow's goldeneye, bufflehead, and cavity-nesters and divers in general (Figures 9 A-D). For divers all of the lake size classes were significant including lakes in the 10-50 hectare class (fwa_10plus). Divers prefer deeper lakes than dabblers and the models appear to capture this dynamic. A deeper exploration of the patterns within the variable importance measures is beyond the scope of this study which is focused on methods and techniques, however, future studies will benefit from a close inspection of these results (see Appendix 6 for dot charts of importance values).

In summary, variable importance values important aids in variable selection, but are not by themselves necessarily interpretive. However, they may reveal causal mechanisms that inform the development of complementary, explanatory models (Shmueli, 2011). Future modeling efforts are advised to clearly formulate objectives at the outset as the aim will determine variable selection and in turn influence predictive accuracy and interpretability.

Table 5. Predictors ranked by relative variable importance for each ecosection averaged over all species-ecosection models, sequentially ordered by average overall rank. Note that some variables were not input into ecosection-specific models and are represented.

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

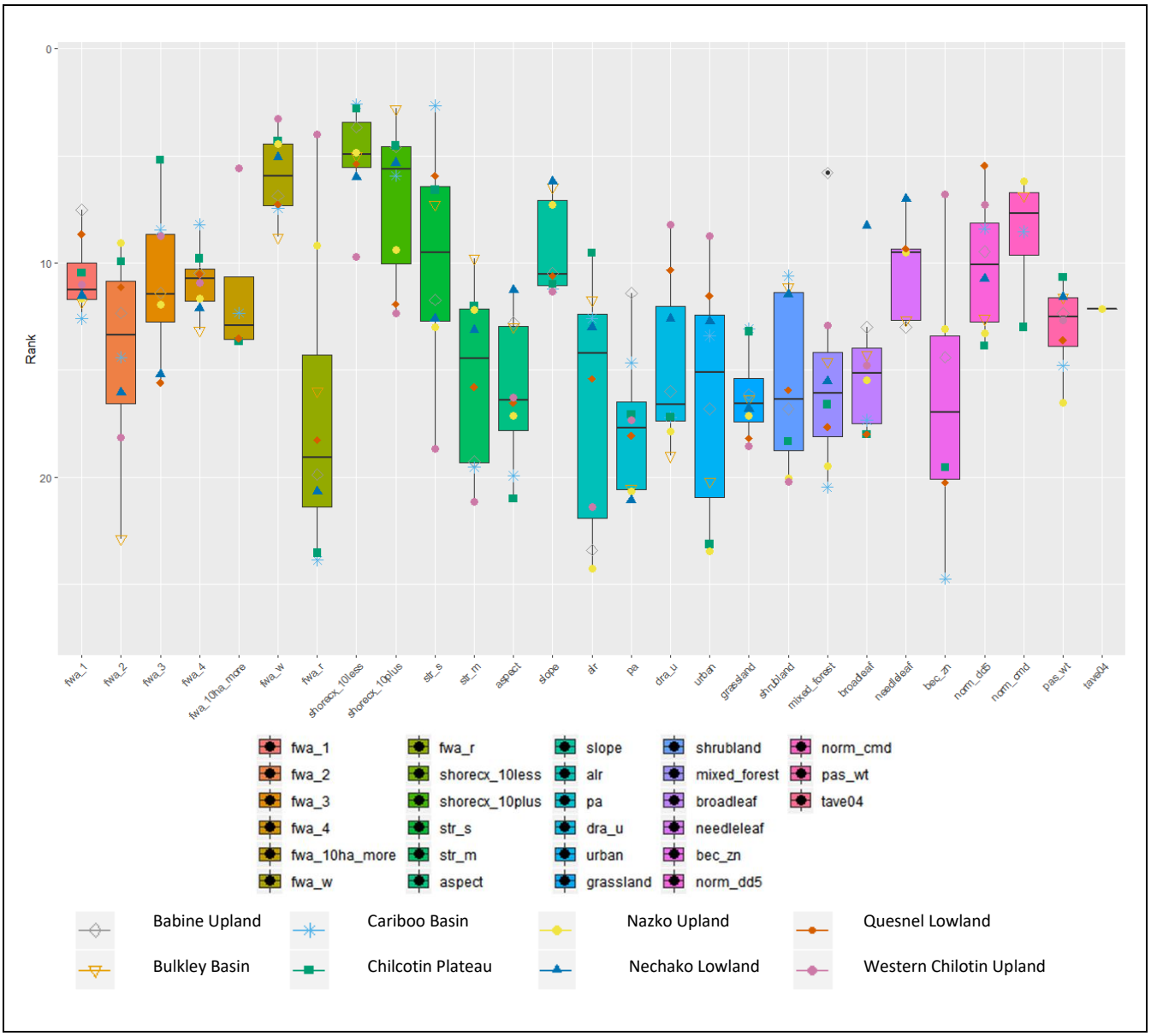


Figure 8. Variable importance rank averaged over all models.

A – American Wigeon (top) and Barrow's Goldeneye (bottom)

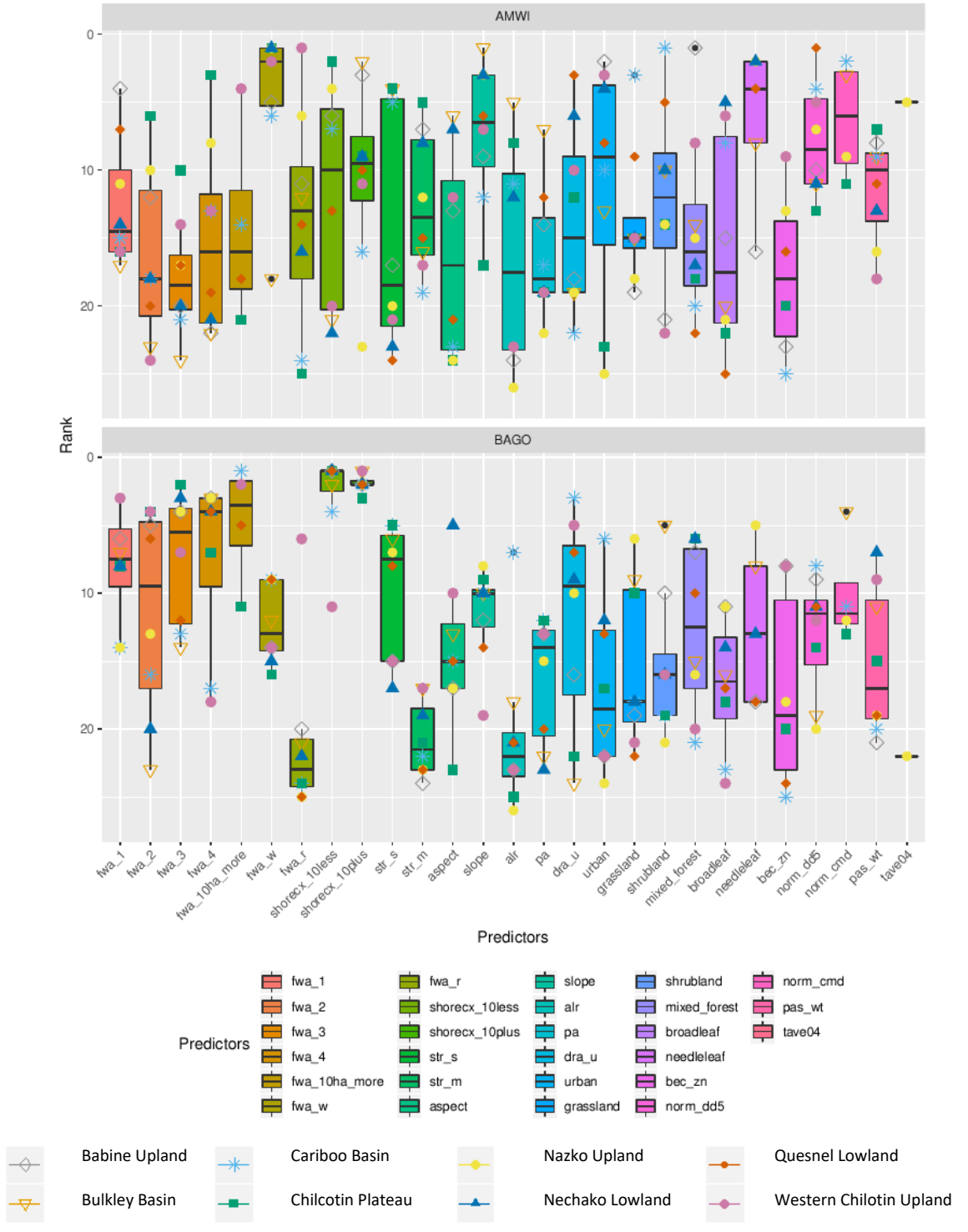


Figure 9A. Predictors ranked by relative variable importance for American wigeon (top) and Barrow's goldeneye (bottom).

B – Bufflehead (top) and Blue-winged Teal (bottom)

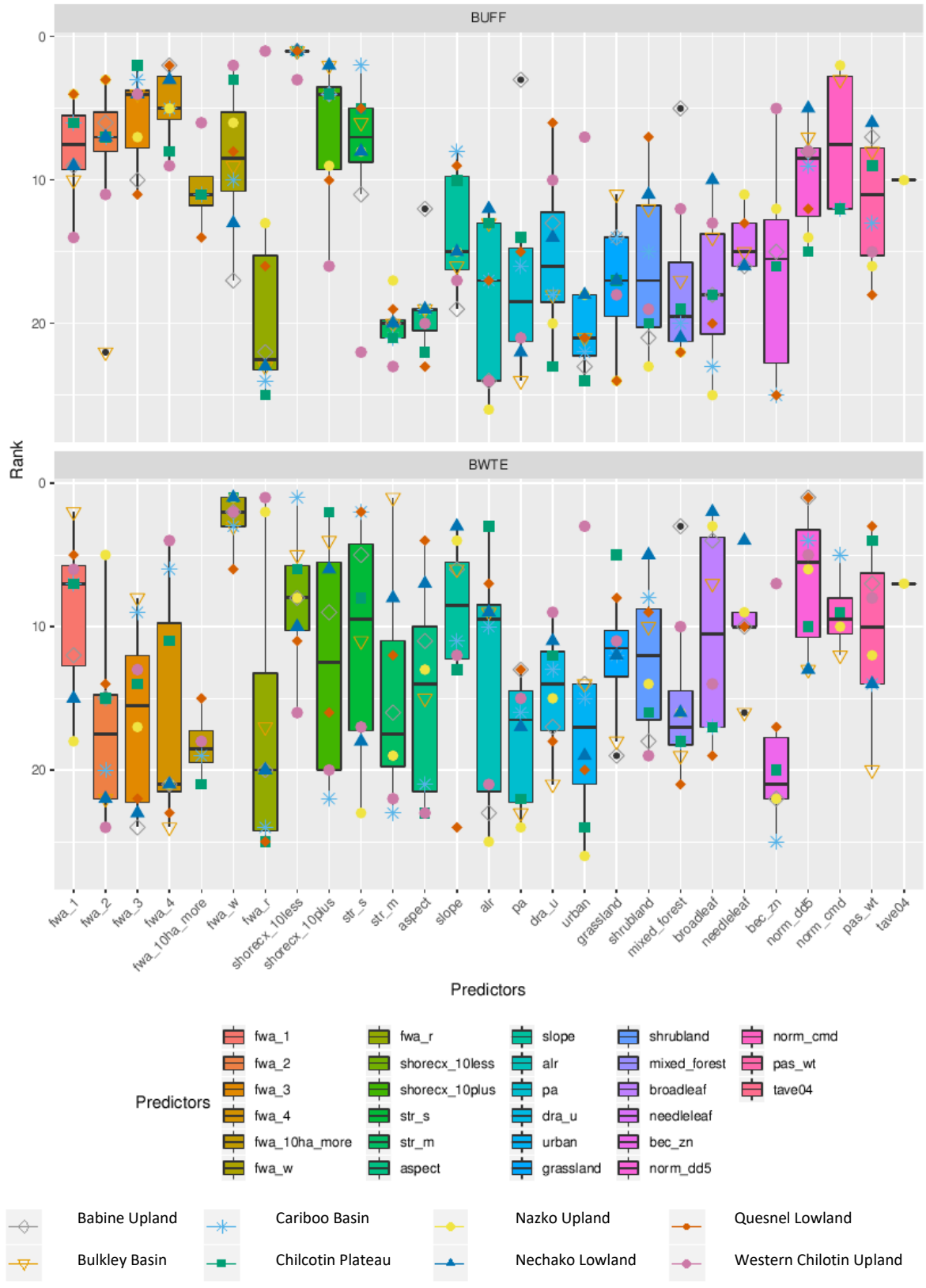


Figure 9B. Predictors ranked by relative variable importance for bufflehead (top) and blue-winged teal (bottom).

C – Canada Goose (top) and Cavity-nesters (bottom)

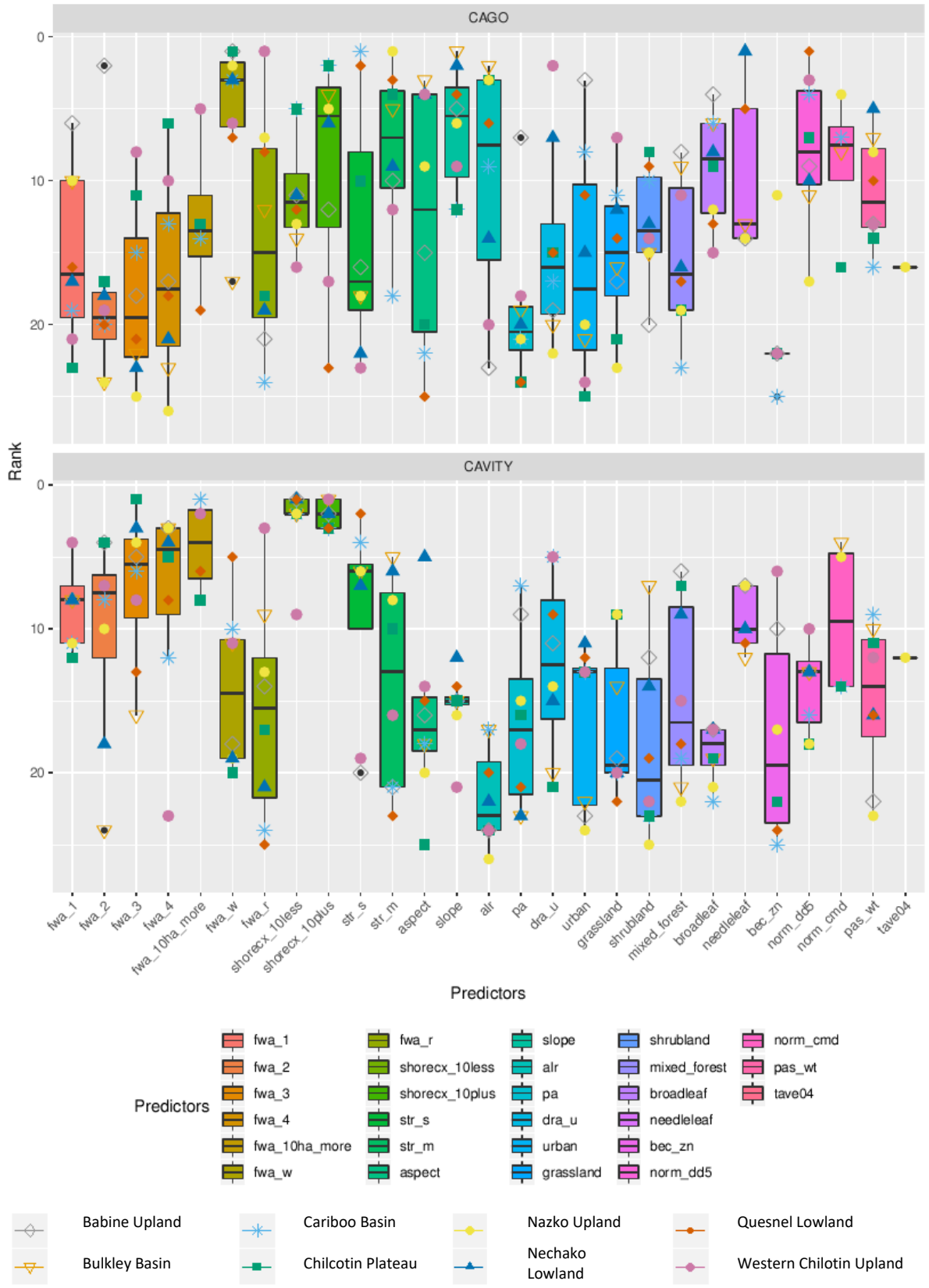


Figure 9C. Predictors ranked by relative variable importance for Canada goose (top) and cavity nesters (bottom).

D – Dabblers (top) and Divers (bottom)

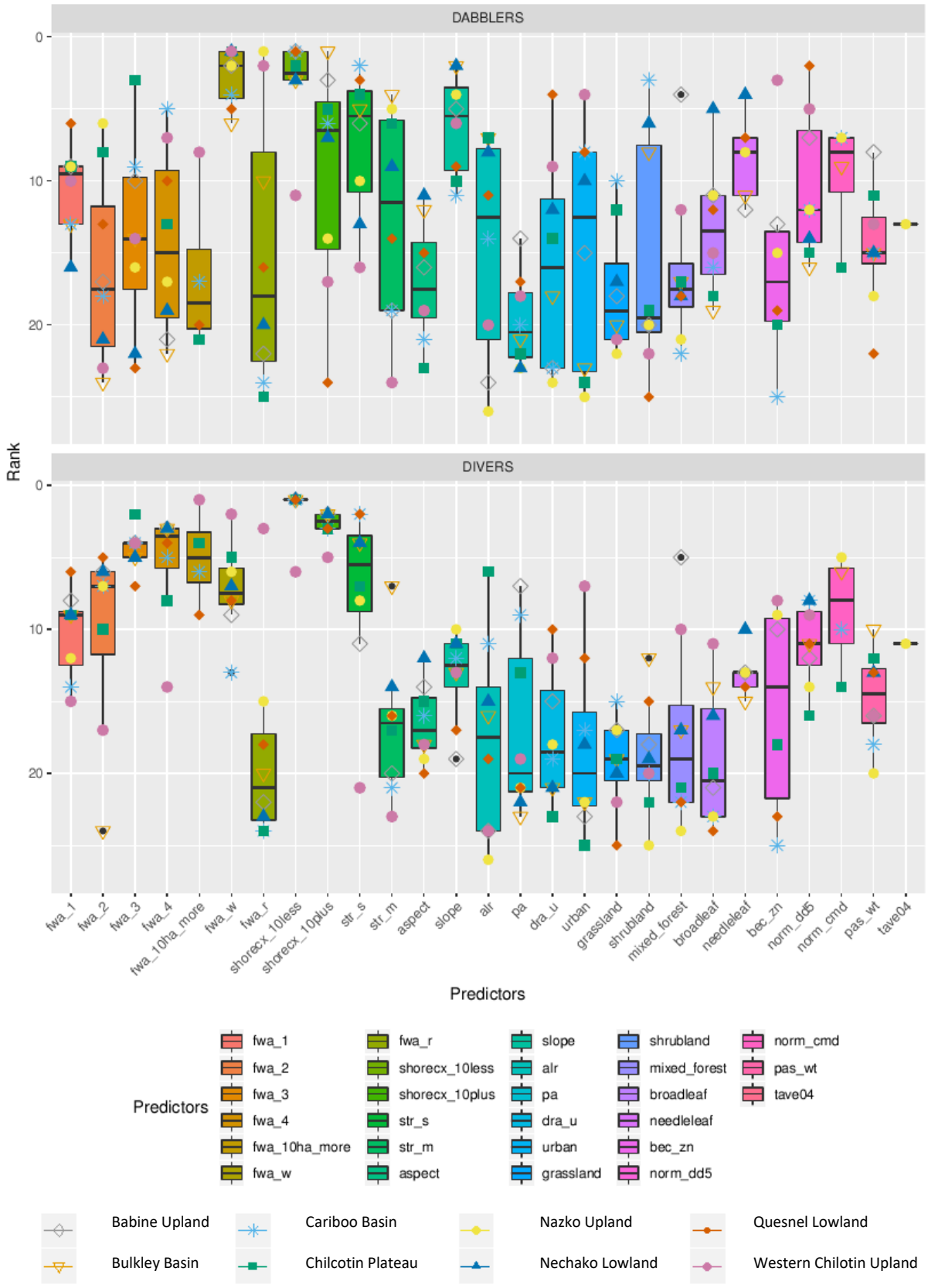


Figure 9D. Predictors ranked by relative variable importance for dabblers (top) and divers (bottom).

E – Green-winged Teal (top) and Mallard (bottom)

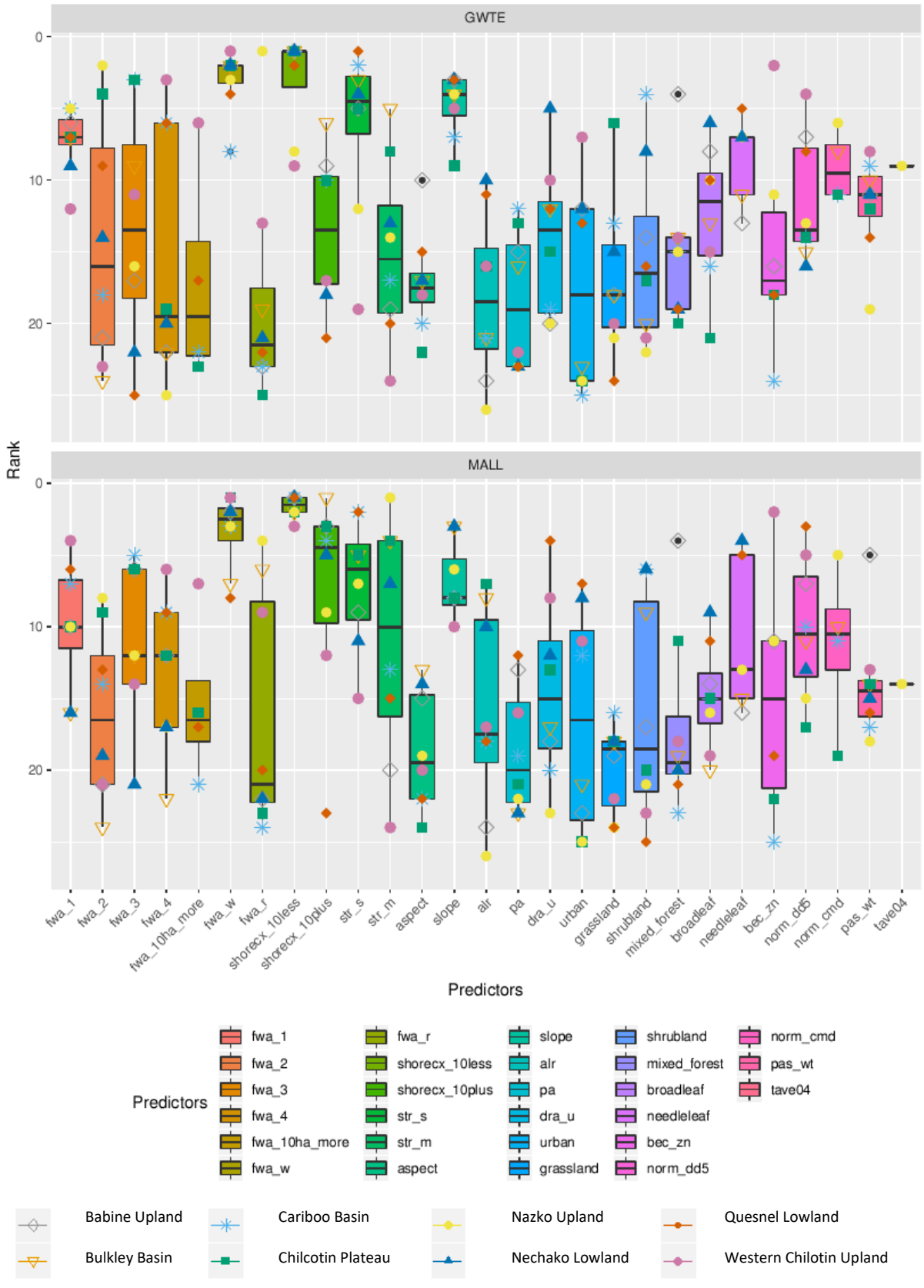


Figure 9E. Predictors ranked by relative variable importance for green-winged teal (top) and mallard (bottom).

F – Northern Shoveler (top) and Ring-necked Duck

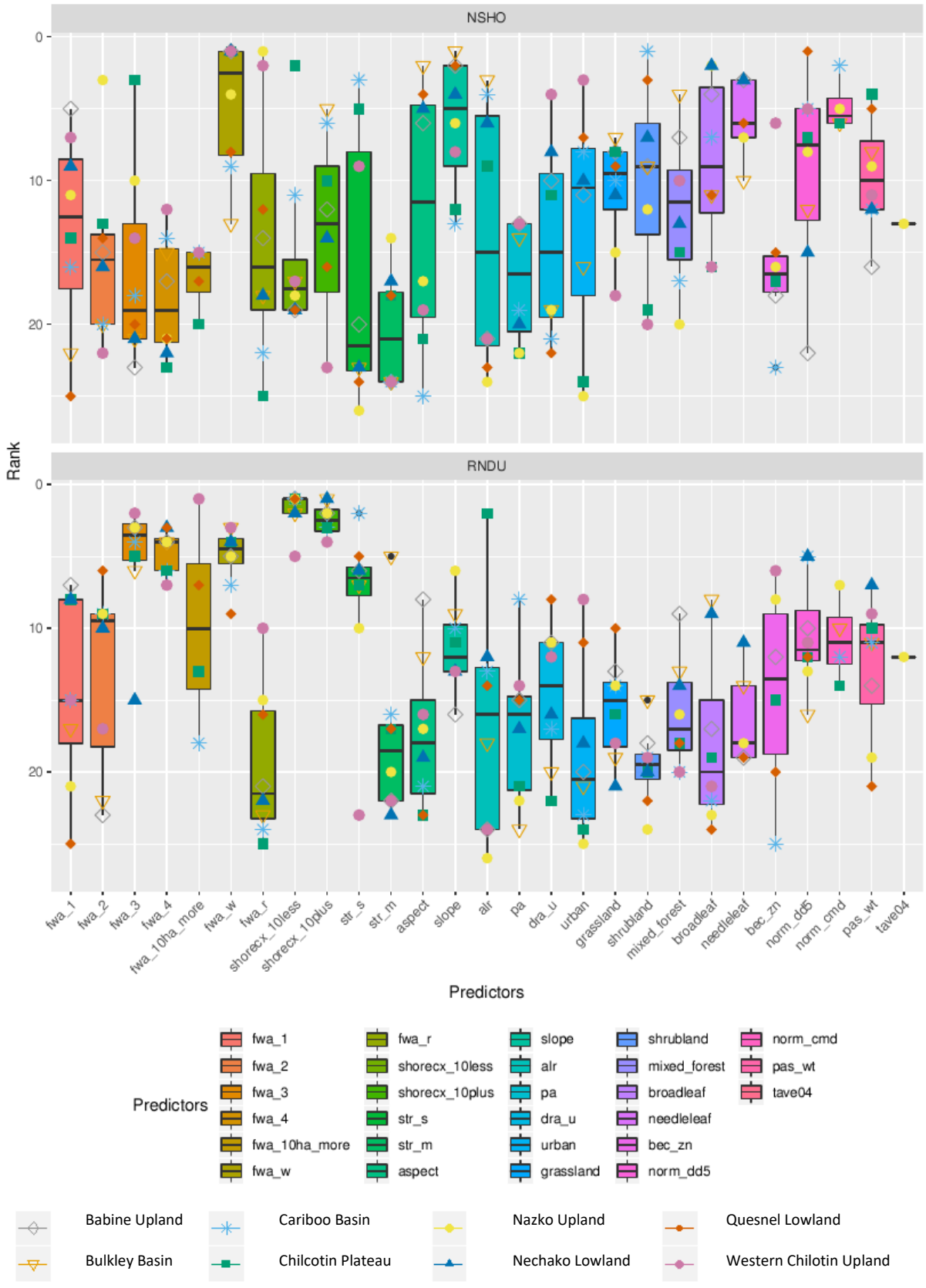


Figure 9F. Predictors ranked by relative variable importance for northern shoveler (top) and ring-necked duck (bottom).

G –Scaup* (top) and Species Richness (bottom)

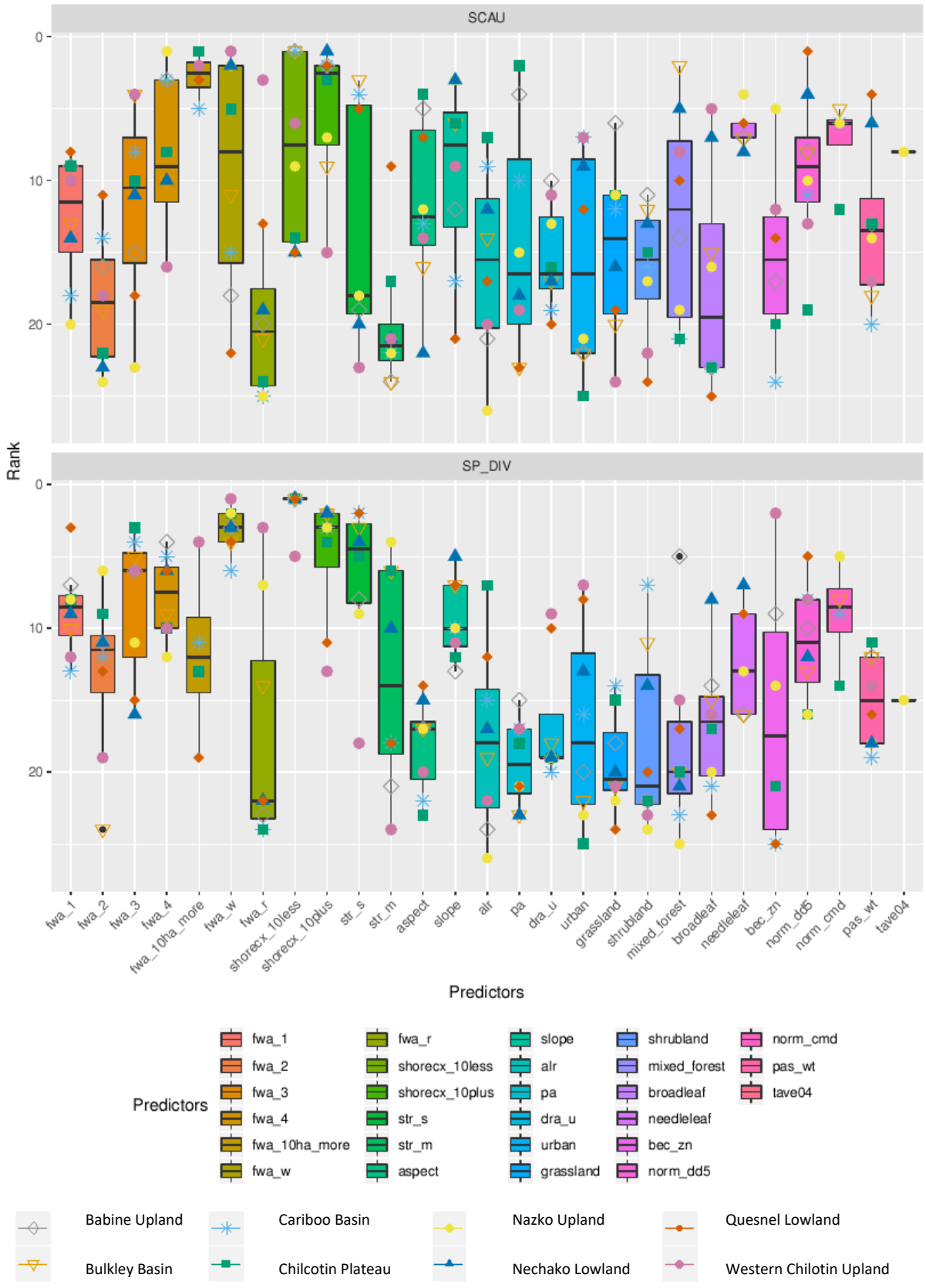


Figure 9G. Predictors ranked by relative variable importance for lesser scaup (top) and species richness (bottom).

* The survey methods do not differentiate between scaup species however only lesser scaup occur in this area.

H – All Species

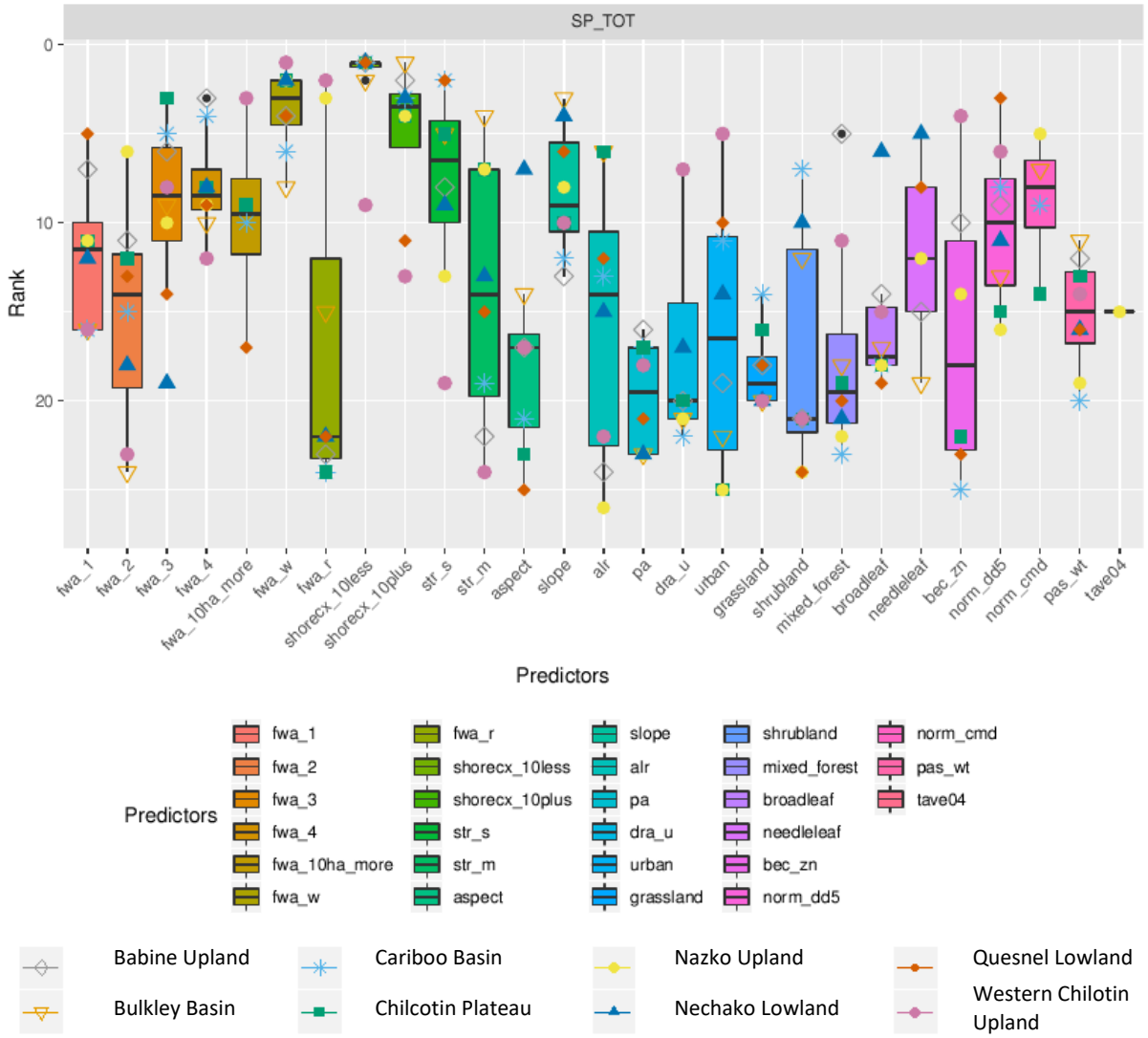


Figure 9H. Predictors ranked by relative variable importance for all species aggregated.

Figure 9. Predictors ranked by relative variable importance for each species (plots A – H). Ecosections are indicated by color and shape).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 7. 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.

	Babine Upland		Bulkley Basin		Cariboo Basin		Chilcotin Plateau		Nazko Upland		Nechako Lowland		Quesnel Lowland		Western Chilcotin Lowland	
Species	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	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	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
Species 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

Overall, the models on average explained 18% of the variation in species distribution (Figure 10). 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.

The MAE averaged over all models was 13%, but this varied widely between species and ecosections (Figure 11). The generalized group classifications excluding cavity-nesters had the largest error values and greatest range in variability between ecosections. The Cariboo Basin consistently scored the highest error rate. High error rates are likely amplified due to the zero-inflation of observation frequencies—birds are not uniformly distributed throughout the landscape, rather most cells have zero counts that are punctuated by hotspots of high abundance. This likely results in lower predicted response values than observed for the more densely populated areas and higher values in the least populous. The internal OOB accuracy measures provide an

objective measure of performance, however it is recommended that model results be evaluated by expert opinion for further assessment and insight.

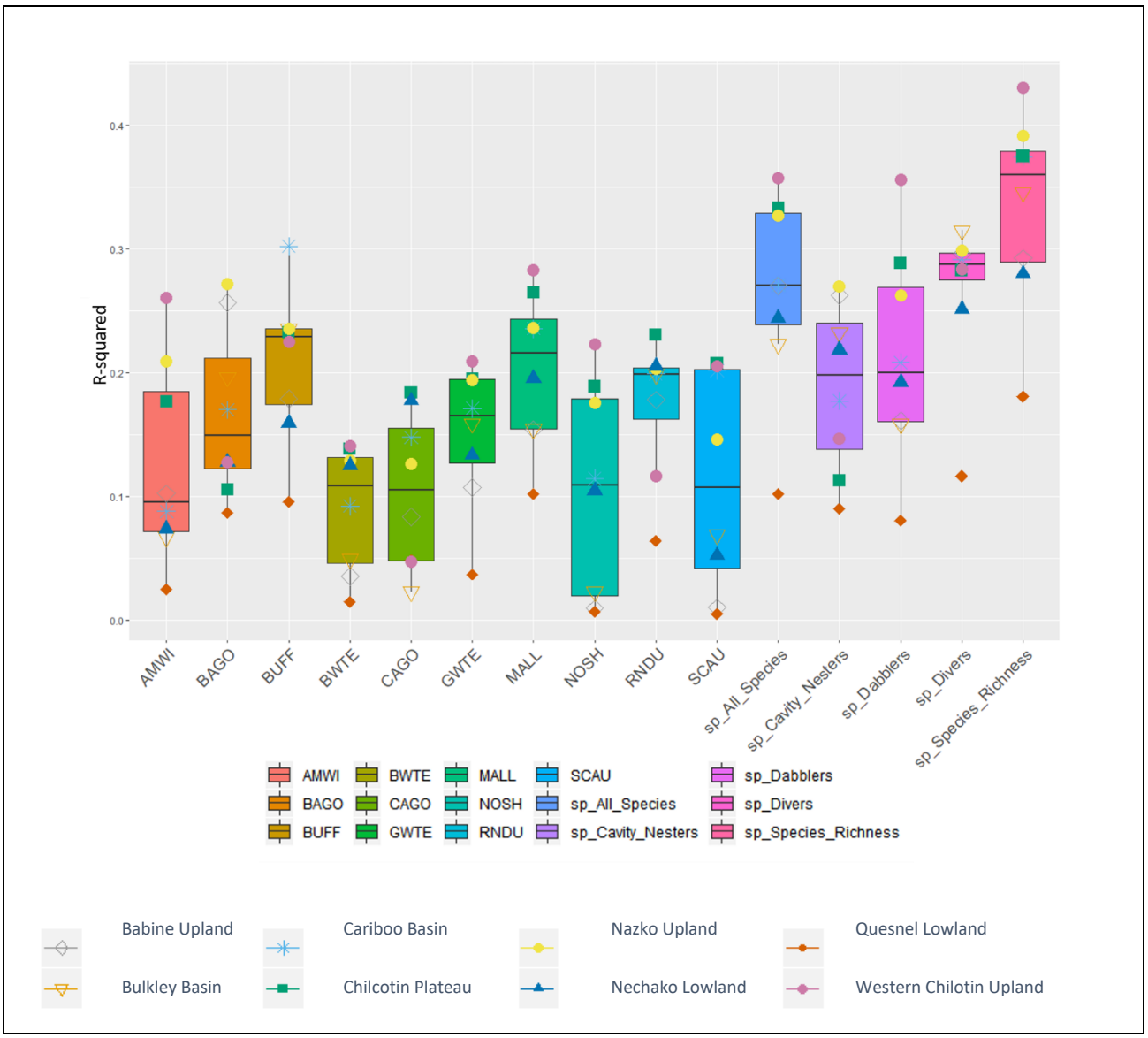


Figure 10. Boxplots of the coefficient of determination (R2) of each species-ecosection model.

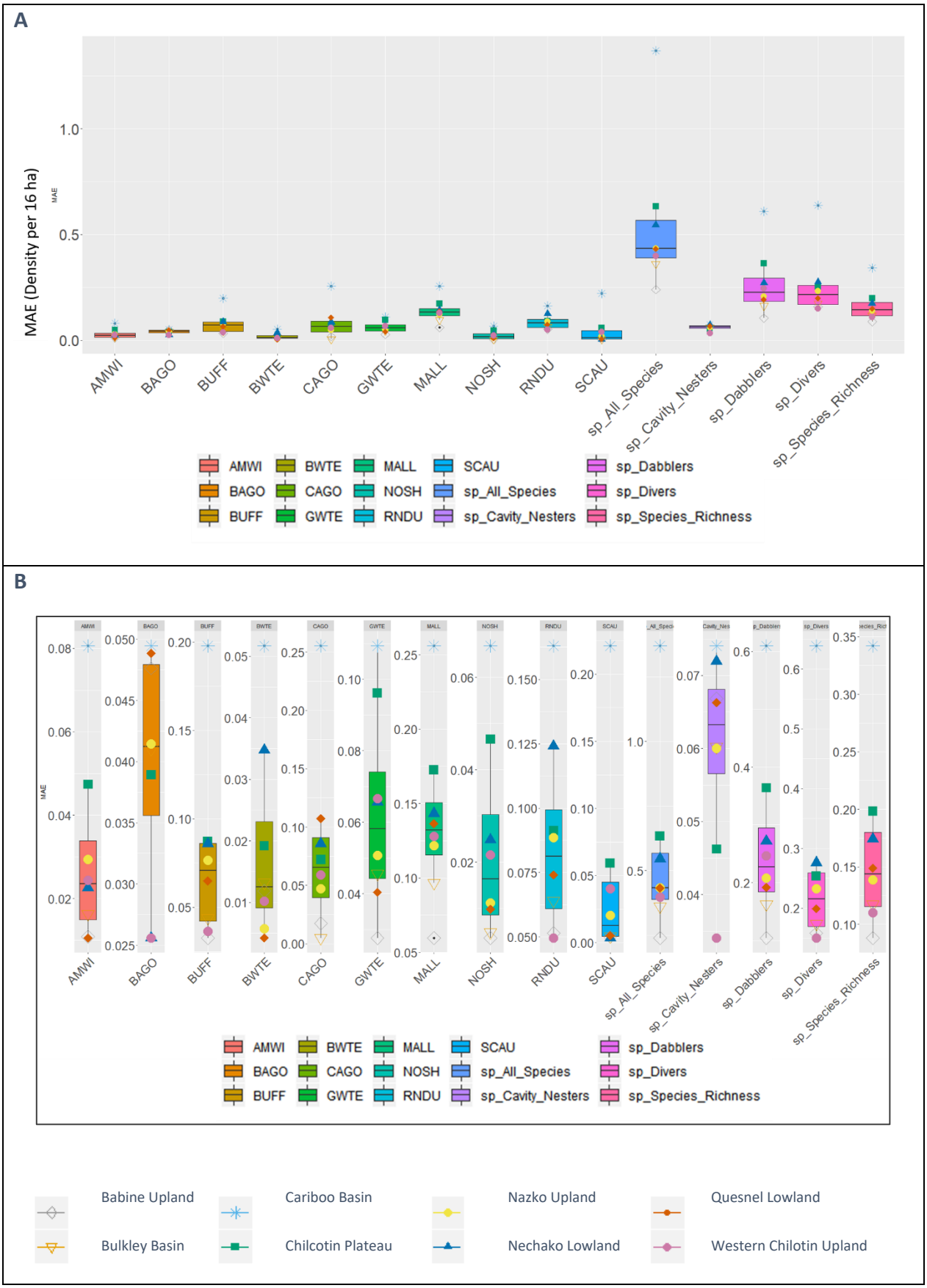


Figure 11. Boxplots of the mean absolute error (MAE) of each species-ecosection specific model (A. Fixed scale, B. dynamic scale range values).

Model Limitations

Data Limitations

It is widely recognized that "ducks like water" (Pimm 1994, Barker et al 2014), 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). Small, ephemeral and seasonal wetlands are often the most productive[2]. Yet these are difficult to capture with freely available, coarser-scale remote sensing imagery. Techniques for finer-scale wetland feature extraction based on Landsat imagery have been developed for open wetlands, but have not yet been fully developed and tested on forested wetlands (Halabisky et al. 2018). Species obligate and facultative preferences for wetlands also vary by wetland class that often occur along smooth, transitional gradients that are difficult to distinguish (Hagy et al. 2014; National Wetlands Working Group 1988). The dearth of high-resolution wetland cover of relevant currency restricted the ability to model the primary habitat type determining species occurrences.

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. The lack of wetland land cover data is widely recognized as a conservation gap that a number of organizations are striving to fill. The CWS is engaged in a collaborative effort to develop methodologies for wetland mapping coincident with the May surveys utilizing Radarsat-2 technologies. Results are promising however the project is still in a preliminary stage and further investigations are ongoing (K. Moore, personal communication, September 30, 2018). Additionally, the Canadian Wetland Inventory project spearheaded by Ducks Unlimited Canada aims to provide a comprehensive national inventory, but BC has not yet been mapped (Canadian Wetland Inventory — Ducks Unlimited Canada, 2019). Global wetland mapping projects include the European Space Agency's Climate Change Initiative which has several exciting projects and remote sensing products of varying temporal and spatial resolution that should be closely 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 we should not delay proactive management measures while waiting for these data.

Hydrological regimes that create and maintain wetland ecosystems are influenced by a host of climatic, geological, physiological and anthropological 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 models may be improved by incorporating snow basin, drainage and/or watershed data. Climate impacts in particulary are notoriously complex. Rather than shy away from the uncertainties, future efforts are advised to take into account the variability between global model predictions and employ ensemble methods that encompass this range in developing adaptive management strategies (R. G. Pike et al. 2010; Spittlehouse and Wang 2016). The hydrology of the central interior is further complicated by the mountain pine beetle outbreak of 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).

[2] The productivity of ephemeral and seasonal wetlands is increased by the presence of aerobic microbes that contribute to nutrient mineralization (Schlesinger and Bernhardt 2013). Small, shallow wetlands permit greater light penetration and provide greater relative shoreline complexity the supports emergent vegetation.

Methods Limitations

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 take into account species life-history such as breeding philopatry, site fidelity, nesting chronology. Nor were species interactions accounted for by factors such as density-dependence or territoriality which . Zero-inflated, negative binomial (ZINB) mixed model methods provide a promising approach to modeling community assemblages as do join-species modeling in detecting species interactions and forecasting accurate predictions (Guisan et al, 2017). 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 relationships between these and other datasets identified in Appendix 1.

The models were developed based on parameters tuned to the Cariboo Basin which as the most populous and wetland-dense ecosections is unique within the study area. Parameter tuning on the remaining ecosections was limited by the project scope and was undertaken mainly for demonstrative purposes. Future studies are encouraged to improve upon the models by thoughtful variable selection and methodical parameter tuning based on clearly articulated study goals as the objective will determine the qualification criteria of parameters and the usability of the results (Guisan and Thuiller 2005). The R package 'caret' is designed specifically for data preprocessing and model generation and contains several machine learning methods including a more traditional random forest implementation. I encourage the exploration of these and other tools in future studies.

Species Abundance and Distribution

Areas of high density largely corresponded to the distribution of hydrological features and aligned with the results of exploratory hot spot analyses (ESRI, ArcGIS Pro v2.3 Hot Spot Analysis(Getis-OrdGi*)). Key differences between models indicated by the variability in predictor importance become visibly apparent in the mapped outputs with pattern variations contrasting at ecosection boundaries (see Appendix 1 for predicted species distribution maps.).

Within the Cariboo Basin there are two areas where the greatest diversity and abundance occur: within the southern half of the ecosection, the central area spanning Flat Lake and Moose Lake Provincial Parks and in the northeastern portion west of the community of Williams Lake and north of McLeese Lake to Quesnel in the Quesnel Lowland ecosection. The areas of high biodiversity in the Chilcotin Plateau occur around Nazko Lake Provincial Park and White Pelican Provincial Park. The pattern of species distributions within Western Chilcotin Upland relative to other ecosections appear over-generalized. The interior region of this ecosection is mainly represented by rounded, high-elevation plateaus with low predicted abundances instead high concentration occur within the lower elevation valleys on the far western edge west of Anahim Lake. In Nazko Upland, the highest densities occur north of Nazko and west of the Telegraph Range within a low elevation depression. In Nechako Lowland the greatest concentrations of waterfowl occur along the Nechako River valley and in and around the lakes and farmlands northwest of Prince George. Within the Bulkley Basin ecosection, the Bulkley River valley from Houston to Smithers within the northwestern arm and from Endako River to Fraser Lake host the greatest relative abundances. And in Babine Upland, the least populous ecosection, the highest concentrations are located around Stuart and Babine Lakes.

Further discussion regarding the mapped SDM results is beyond the scope of this study but a deeper inspection and exploration of the patterns and relationships is encouraged (see Figures 12 and 13 for predicted total abundance and average densities for the top ten most abundant species).

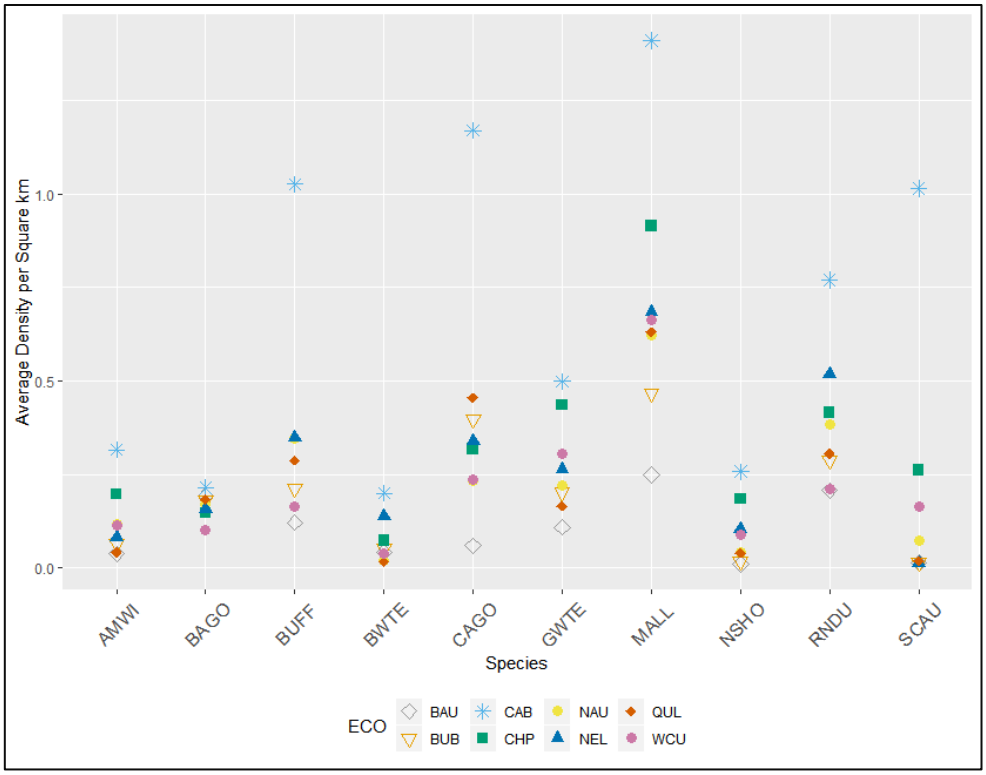


Figure 12. Predicted average density per square kilometre of total indicated breeding population.

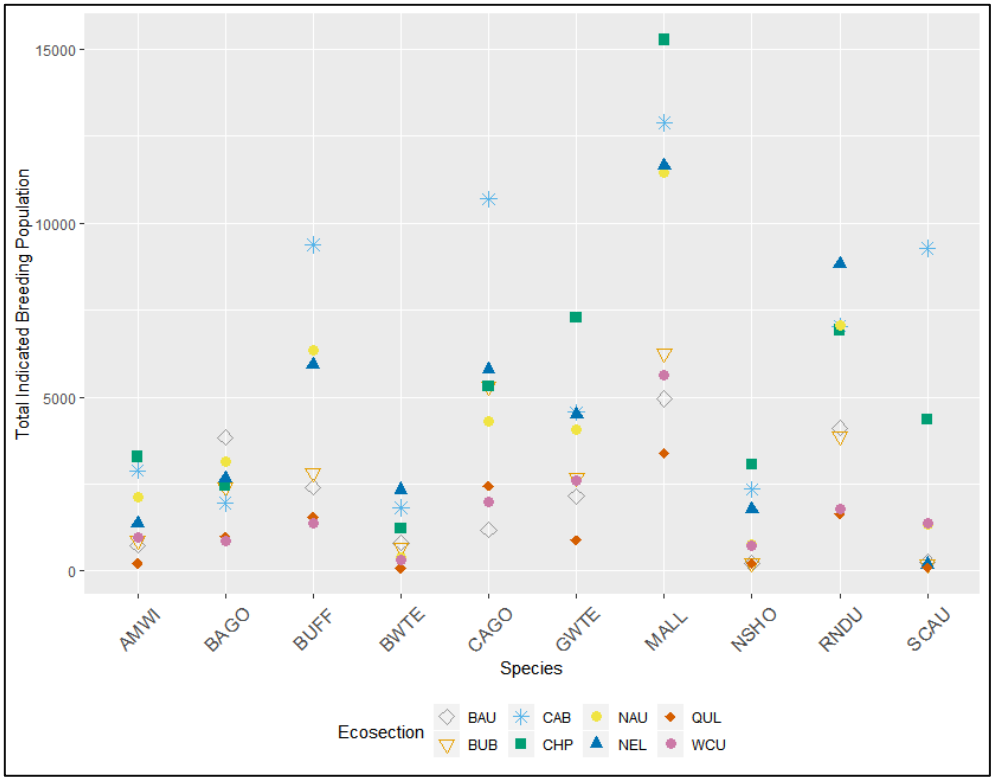


Figure 13. Predicted total indicated breeding population for the top ten most abundant species for each ecosection.

Applications

This modeling study had two primary, related objectives: first, develop the methods and techniques required to standardize the BC May Surveys for analysis and second, create SDMs of relative abundance and distribution to support conservation planning and habitat management. The predicted distribution maps can be used for any application requiring an understanding of waterfowl species' use of space such as assessing protection measures and tenure of areas of high abundance and diversity, providing finer-scale population estimates for impact assessments, or in more general terms, to inform future studies of ecological relationships. Most importantly, the results can serve as guidance for the development of SDMs that address the study's limitations and are constructed for targeted conservation action.

A key conservation challenge is to mitigate the impacts of climate change. Changes in BC's hydrological regimes due to increased warming and drying trends (Ministry of Environment 2016) are predicted to lead to wetland losses at low to mid-elevations (Bunnell, Fred L, Wells, Ralph 2010). As higher elevations are less sensitive to temperature changes that can affect snowpack accumulation, these 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 1,200 metres (Bunnell, Fred L, Wells, Ralph 2010), may increase 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). SDMs can be developed to explicitly model the projected impacts and help inform adaptive management measures.

CONCLUSION

This study was undertaken to demonstrate the power and utility of SDMs to guide and support place-based decision-making and science-based conservation action. Models must be developed and communicated in meaningful ways to be useful. Future studies are encouraged to validate and build upon these results by exploring and employing best available data and methods. Encouraging technological developments include alternative, community-based population models that take into account species interactions (Guisan, Wilfried, and Zimmermann, 2017) and the development of remote sensing techniques and data products to delineate wetlands by season and type (Gabrielsen, Murphy, and Evans 2016; Halabisky et al. 2018; Mahdavi et al. 2018). Models are useful only when they bring together expert opinion, ecological principles and systematic monitoring and research in a comprehensive framework that leads to conservation action (Sinclair et al, 2010). It is hoped this study will stimulate future coordinated efforts by CWS, governmental and other partners to better understand species' distributions through applied GIS and SDMs.

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.

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.

Gabrielsen, Charlotte G., Melanie A. Murphy, and Jeffrey S. Evans. 2016. "Using a Multiscale, Probabilistic Approach to Identify Spatial-Temporal Wetland Gradients." *Remote Sensing of Environment*.

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.

Hagy, Heath M. et al. 2014. "Wetland Issues Affecting Waterfowl Conservation in North America." *Wildfowl* (December): 343–67.

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.

Mahdavi, Sahel et al. 2018. "Remote Sensing for Wetland Classification: A Comprehensive Review." *GIScience & Remote Sensing* 55(5): 623–58. https://www.tandfonline.com/doi/full/10.1080/15481603.2017.1419602 (August 10, 2019).

McKenney, Daniel W. et al. 2011. "Customized Spatial Clinate Models for North America." *Bulletin of the American Meteorological Society* (December): 1611–22.

Ministry of Environment. 2016. *Climate Change Indicators for British Columbia 2016 Update*.

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.

Ministry of Forests, Lands, Natural Resource Operations and Rural Development (Organization/Institution). 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).

National Wetlands Working Group. 1988. *Wetlands of Canada*. Sustainable Development Branch, Environment Canada.

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).

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

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." :

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 – Species Distributions Maps

AMWI - American Wigeon

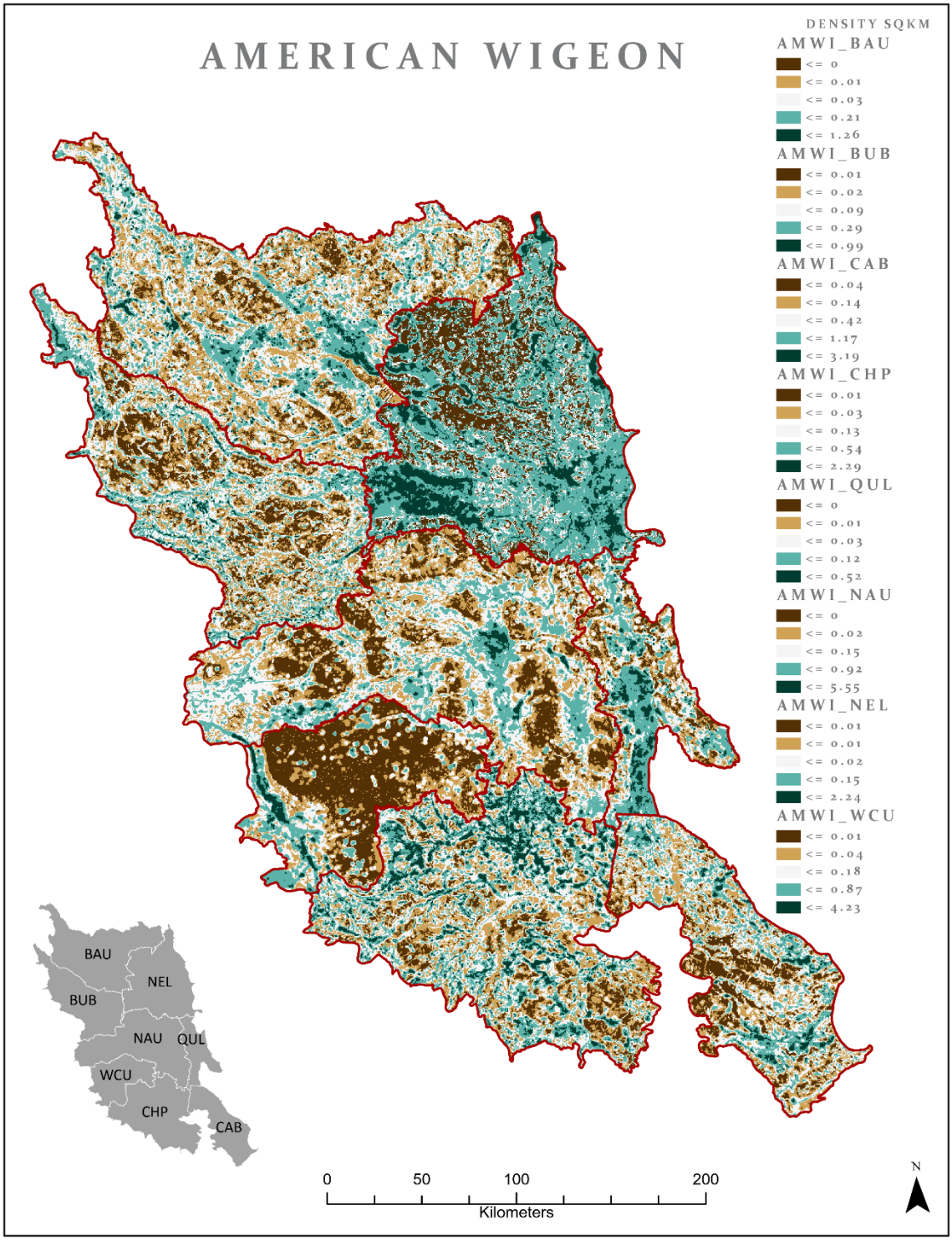


Figure A1-1. Population density per square kilometre forecast for American wigeon. Density classes have been mapped according to a geometric classification for each ecosection.

BAGO - Barrow's Goldeneye

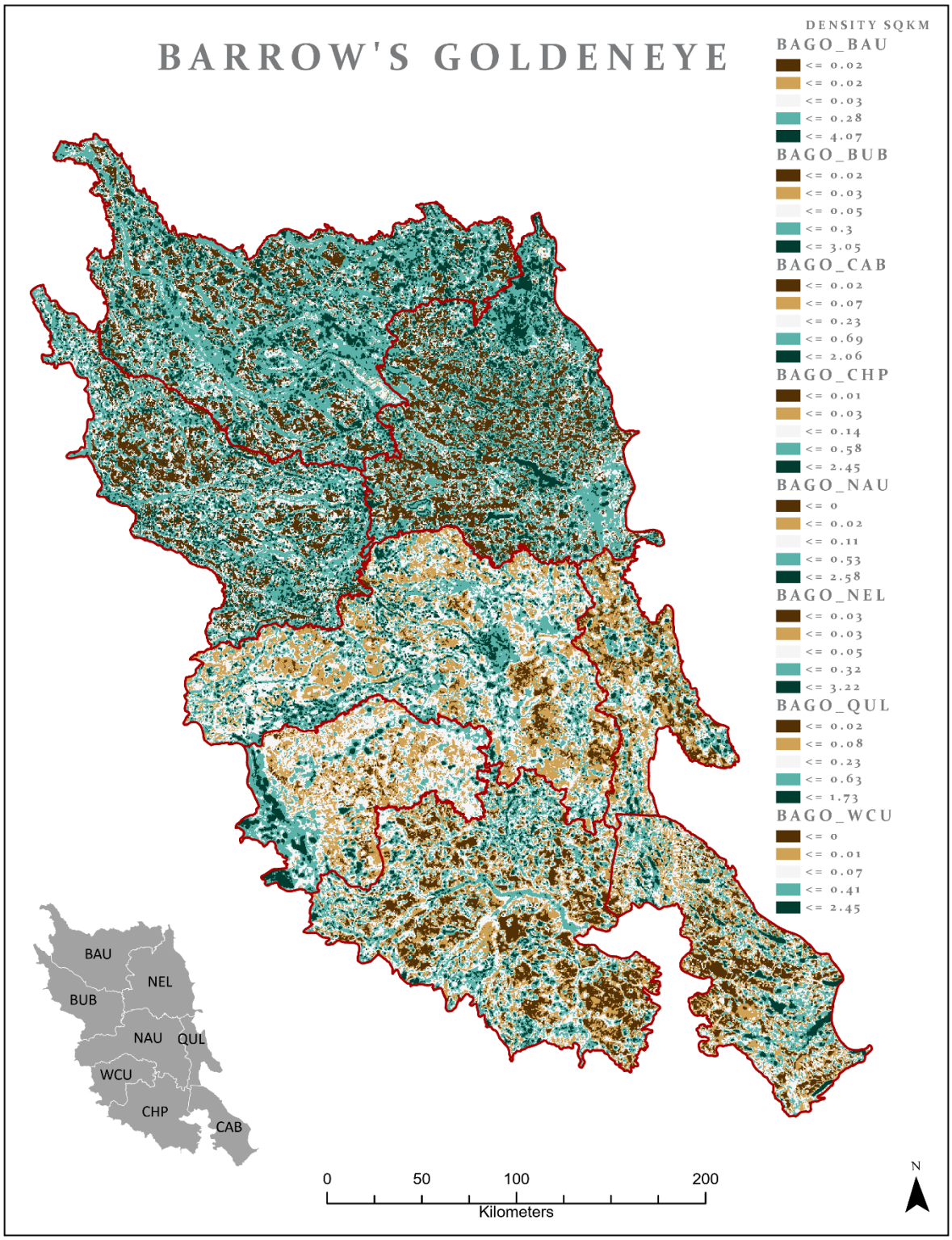


Figure A1-2. Population density per square kilometre forecast for Barrow's goldeneye. Density classes have been mapped according to a geometric classification for each ecosection.

BWTE - Blue-winged Teal

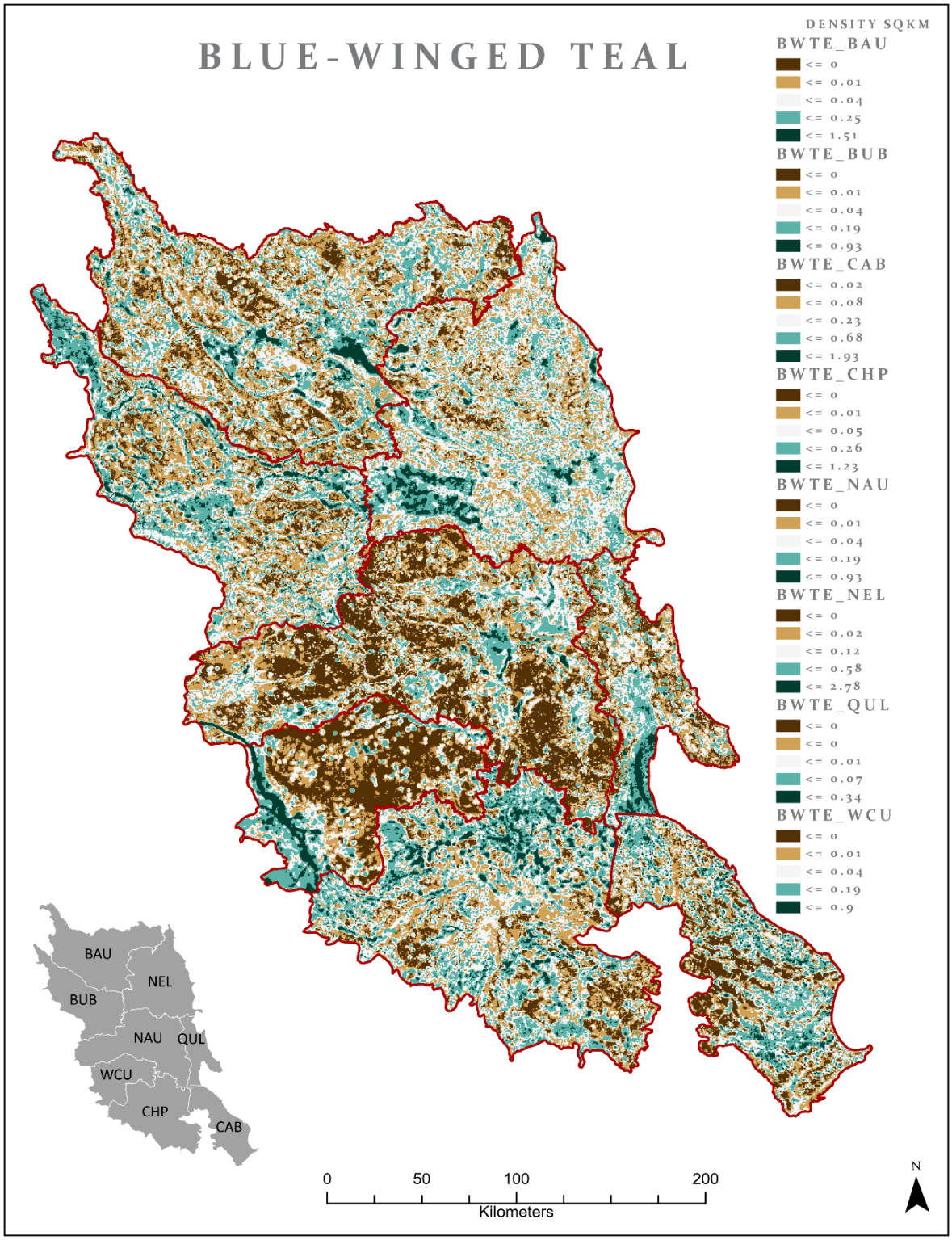


Figure A1-3.Population density per square kilometre forecast for blue-winged teal. Density classes have been mapped according to a geometric classification for each ecosection.

BUFF – Bufflehead

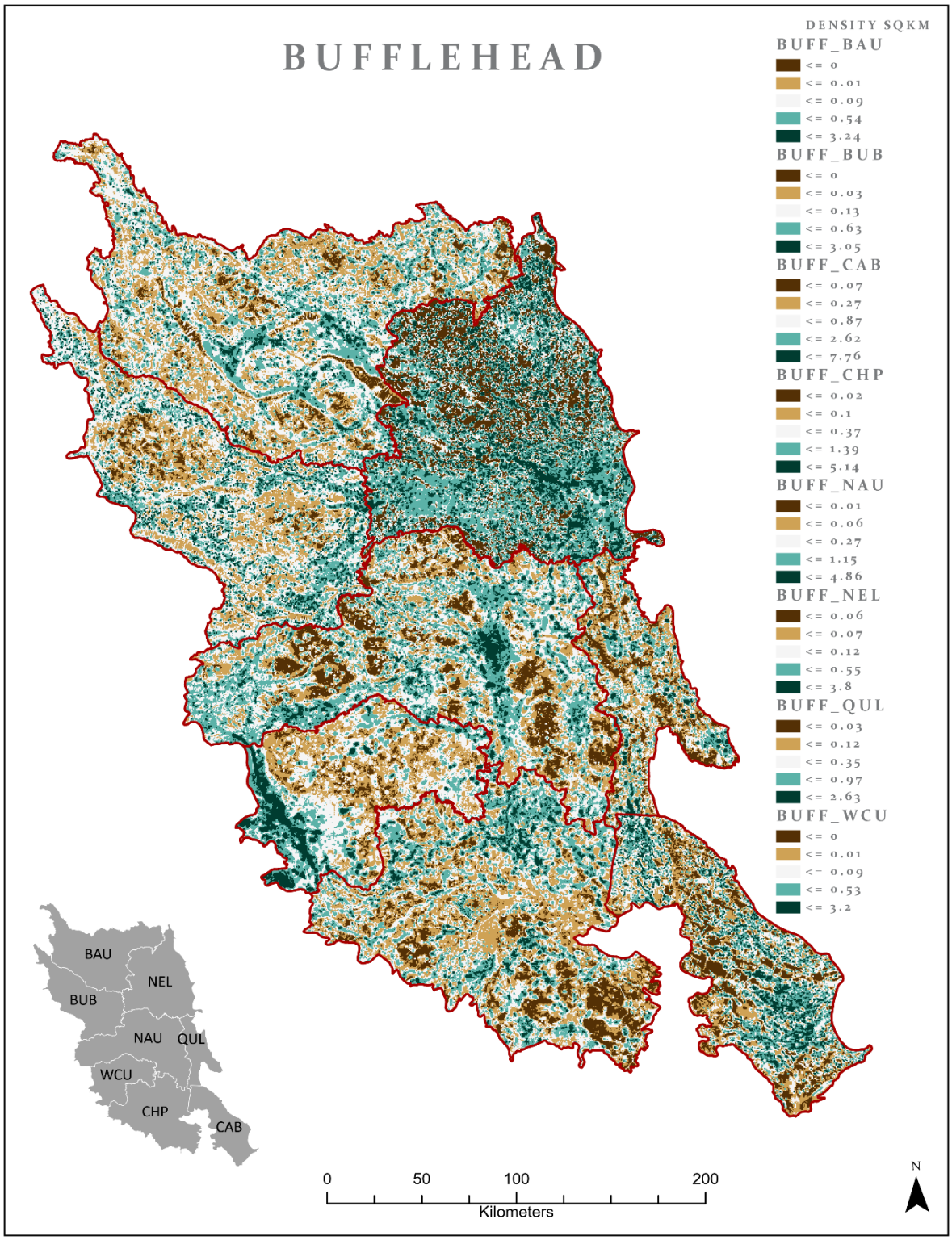


Figure A1-4. Population density per square kilometre forecast for bufflehead. Density classes have been mapped according to a geometric classification for each ecosection.

CAGO - Canada Goose

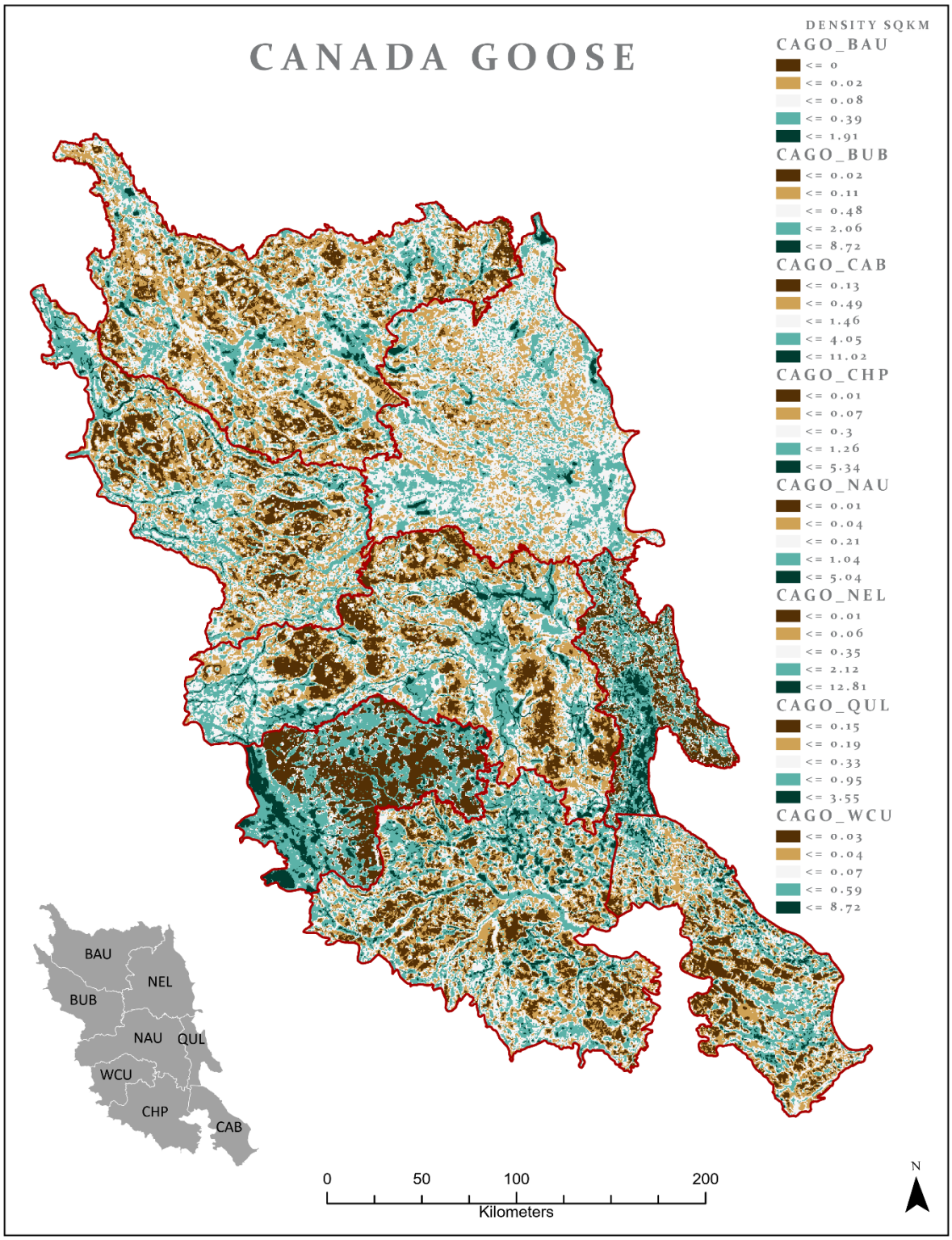


Figure A1-5. Population density per square kilometre forecast for Canada goose. Density classes have been mapped according to a geometric classification for each ecosection.

GWTE - Green-winged Teal

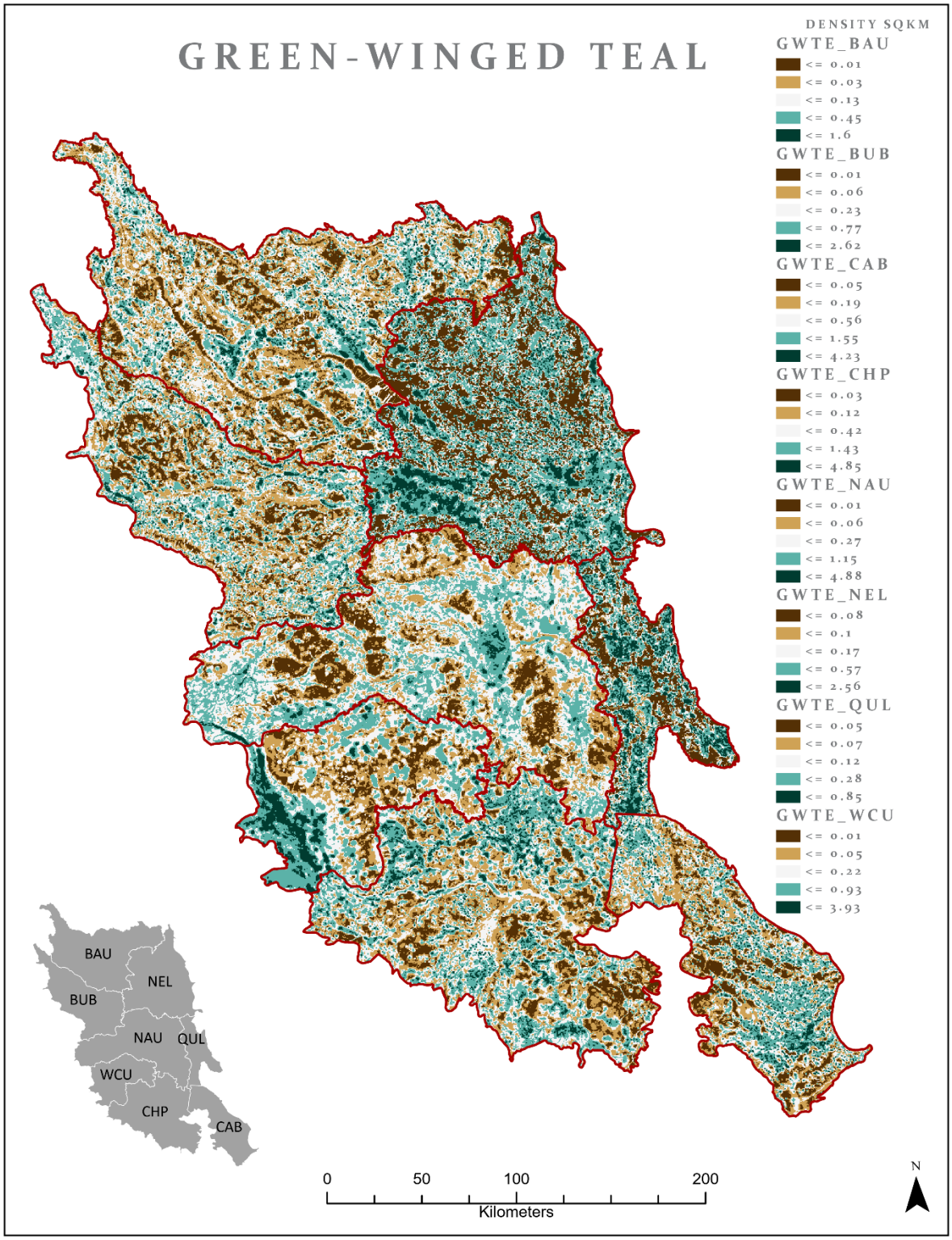


Figure A1-6. Population density per square kilometre forecast for green-winged teal. Density classes have been mapped according to a geometric classification for each ecosection.

MALL – Mallard

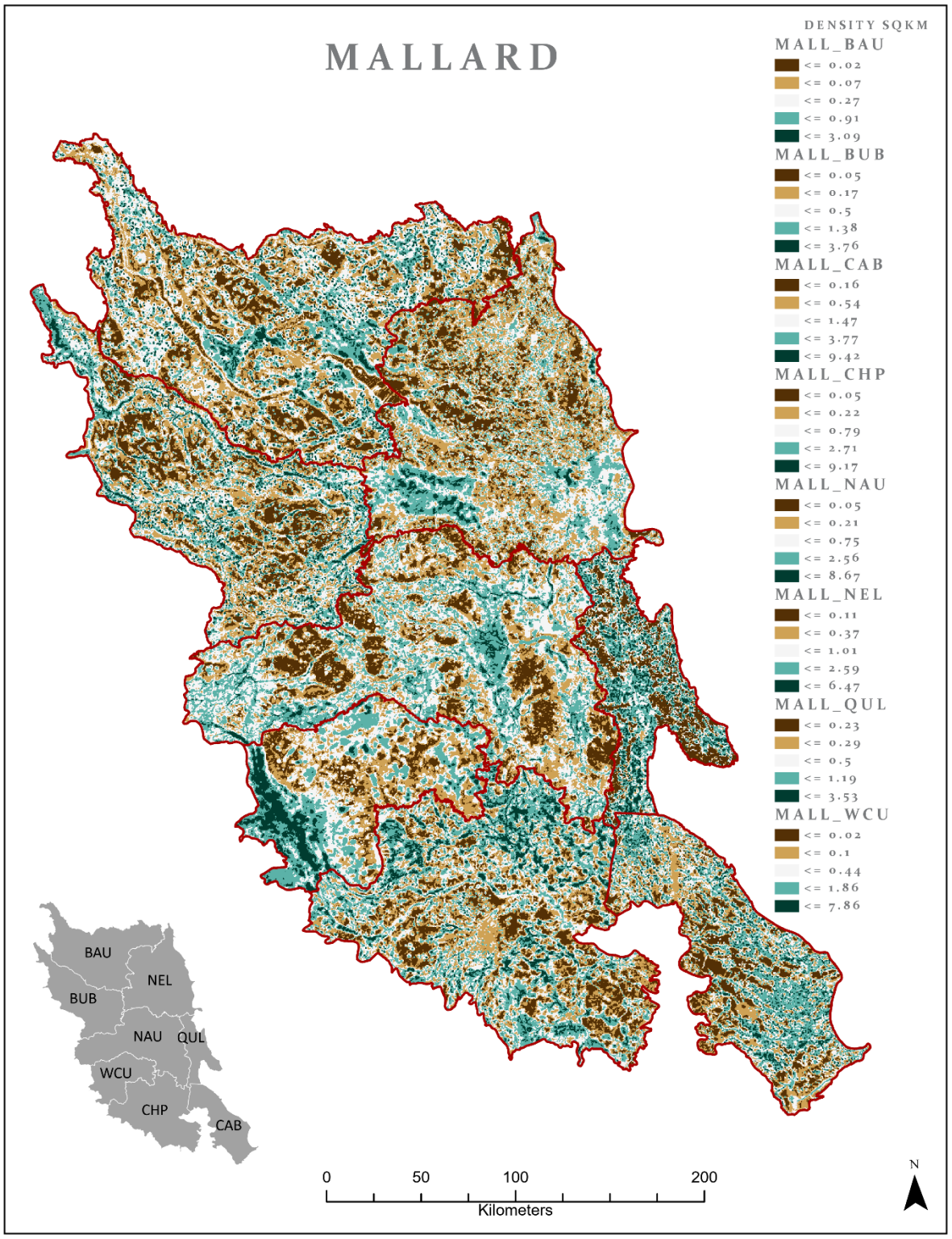


Figure A1-7. Population density per square kilometre forecast for mallard. Density classes have been mapped according to a geometric classification for each ecosection.

NOSH - Northern Shoveler

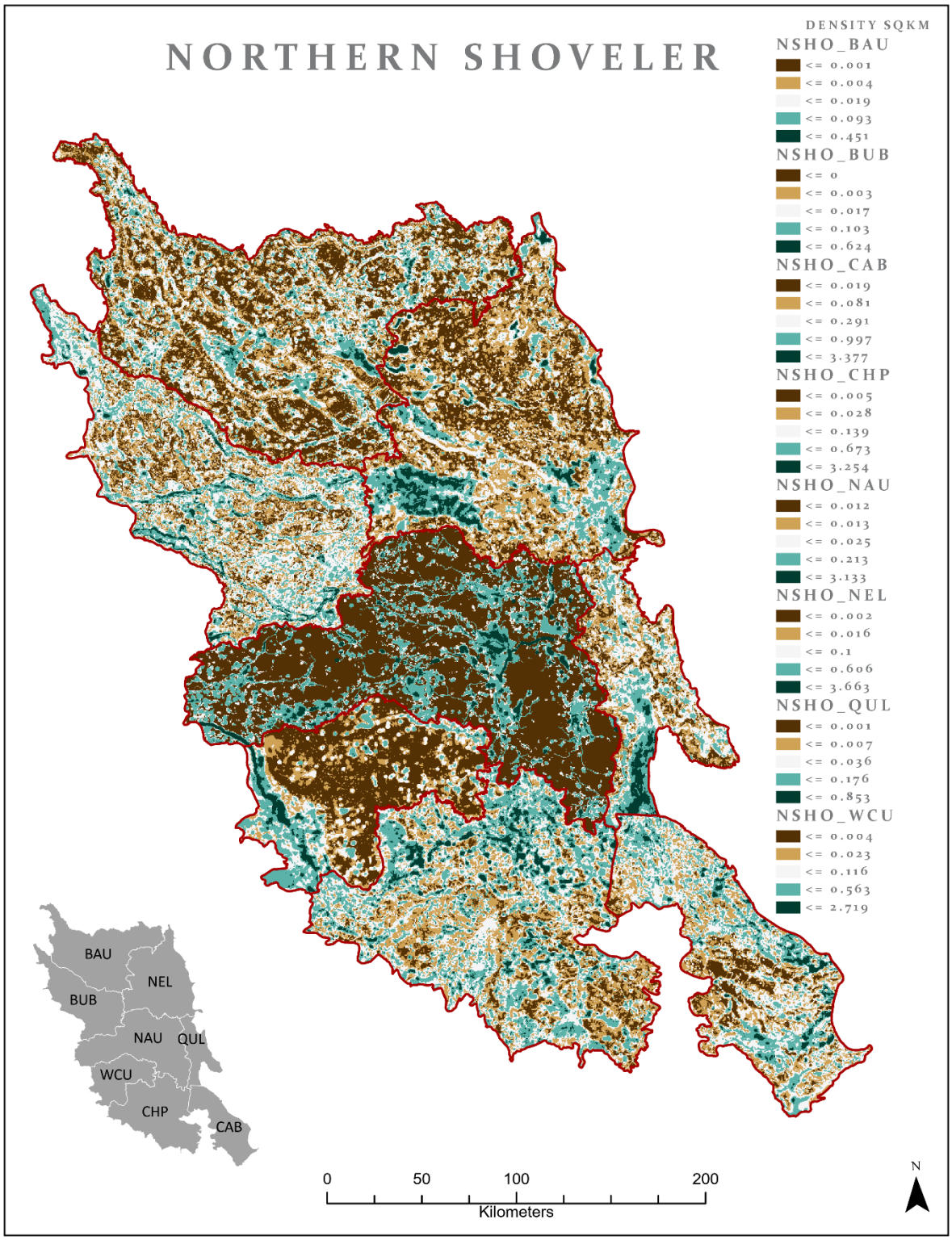


Figure A1-8. Population density per square kilometre forecast for northern shoveler. Density classes have been mapped according to a geometric classification for each ecosection.

RNDU – Ring-necked Duck

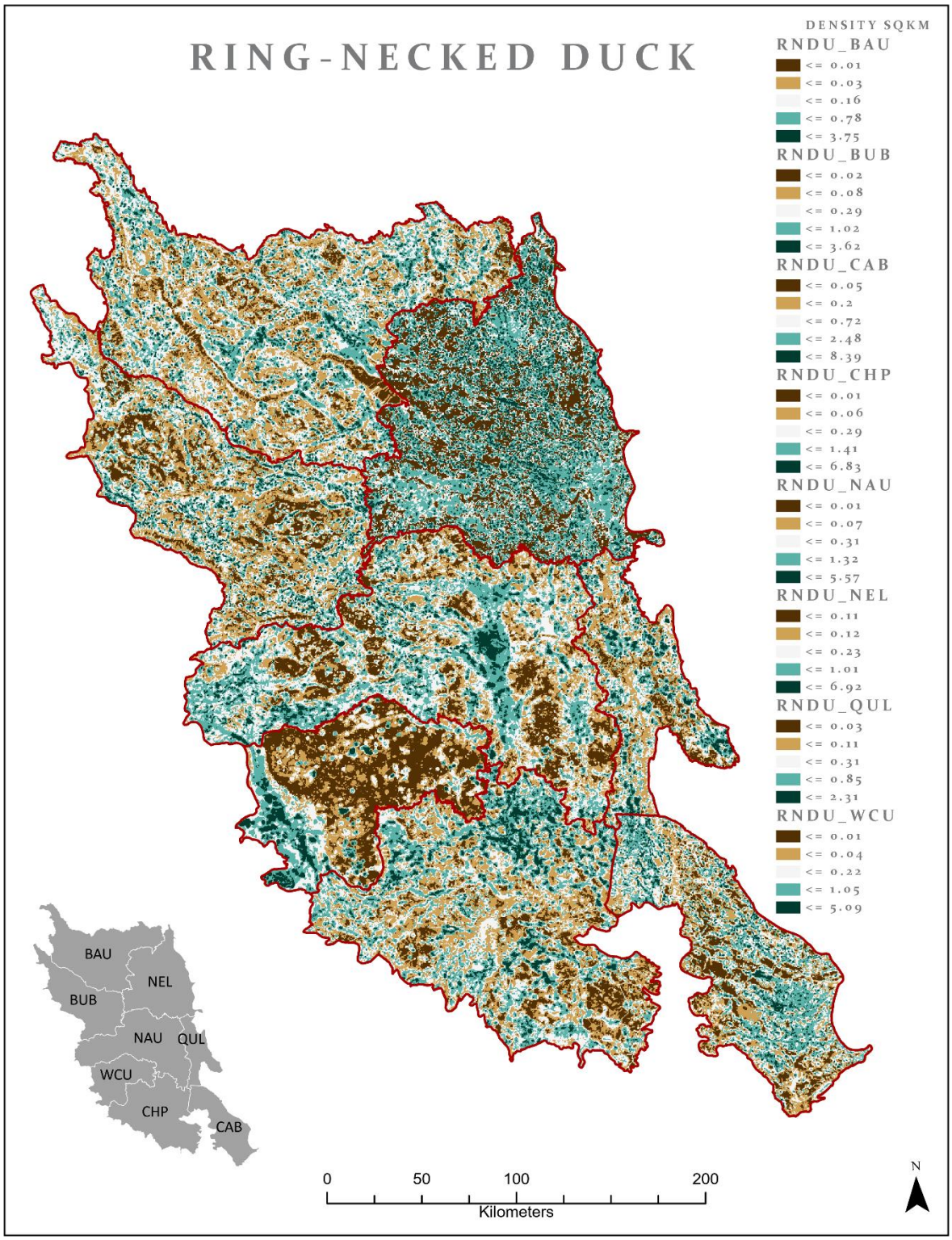


Figure A1-9. Population density per square kilometre forecast for ring-necked duck. Density classes have been mapped according to a geometric classification for each ecosection.

SCAU – Generic Scaup (Lesser Scaup)*

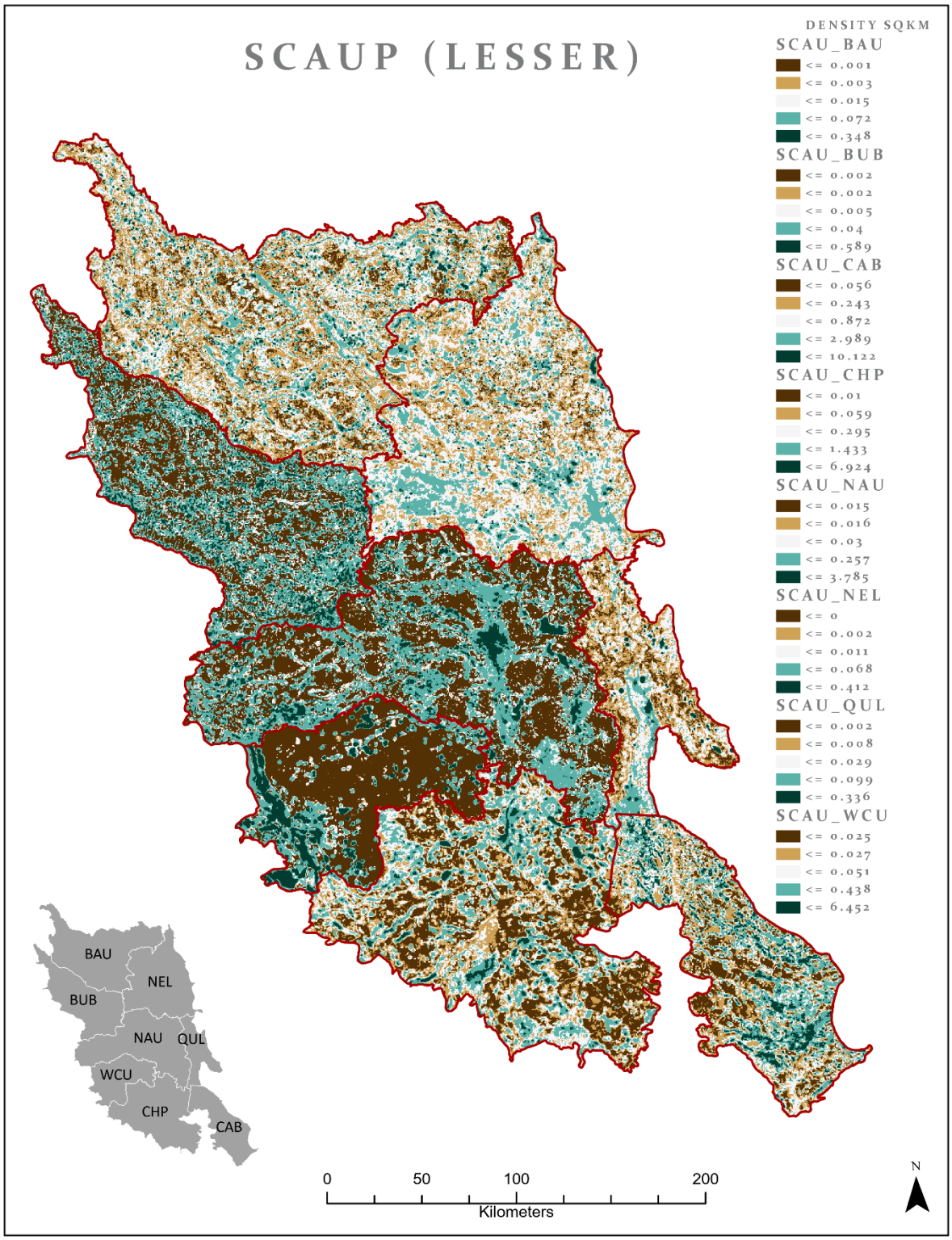


Figure A1- 10. Population density per square kilometre forecast for scaup*. Density classes have been mapped according to a geometric classification for each ecosection.

* The BC May surveys do not distinguish scaup species but only lesser scaup occur in this region.

CAVITY – Cavity nesters

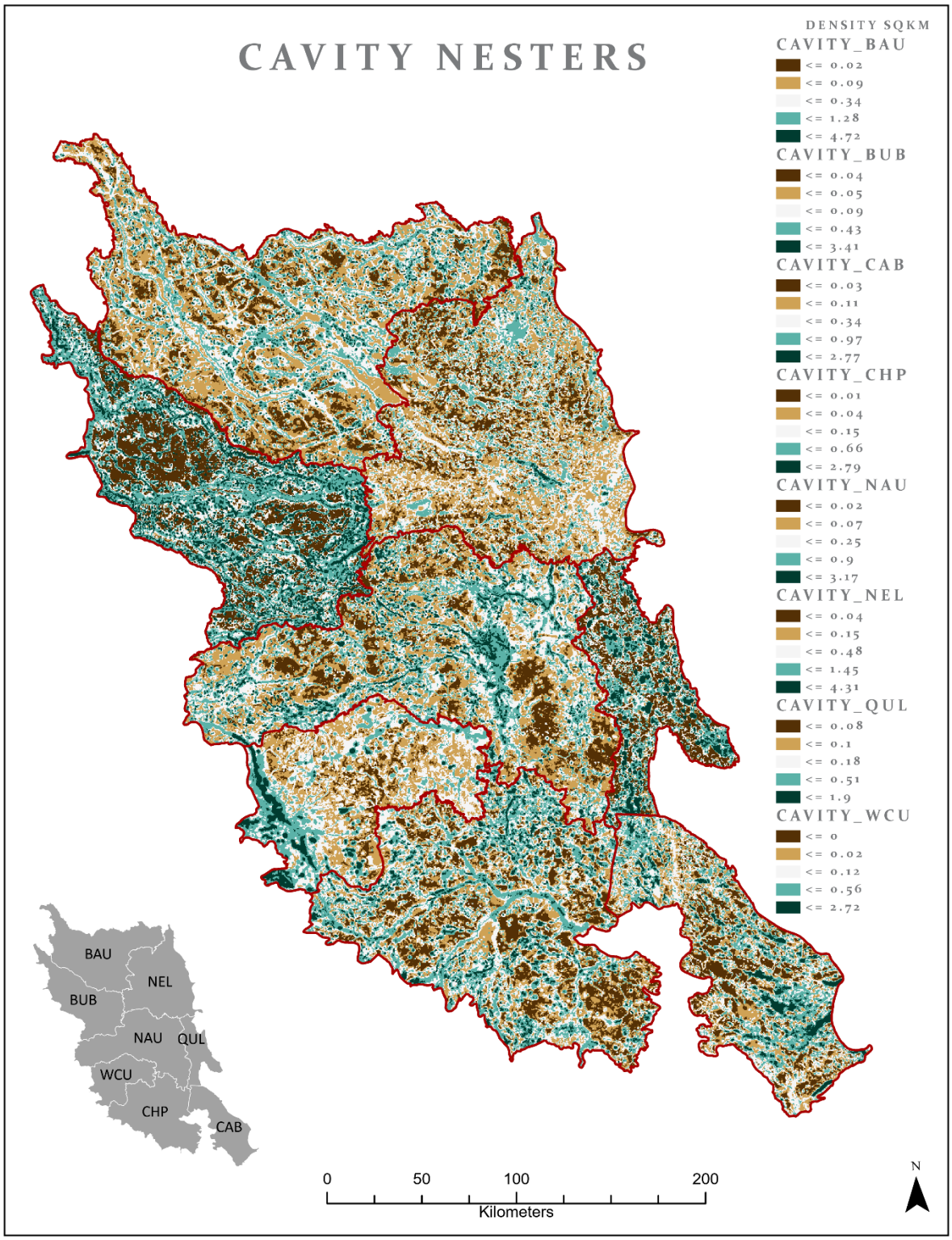


Figure A1-11. Population density per square kilometre forecast for cavity nesters. Density classes have been mapped according to a geometric classification for each ecosection.

DABBLERS

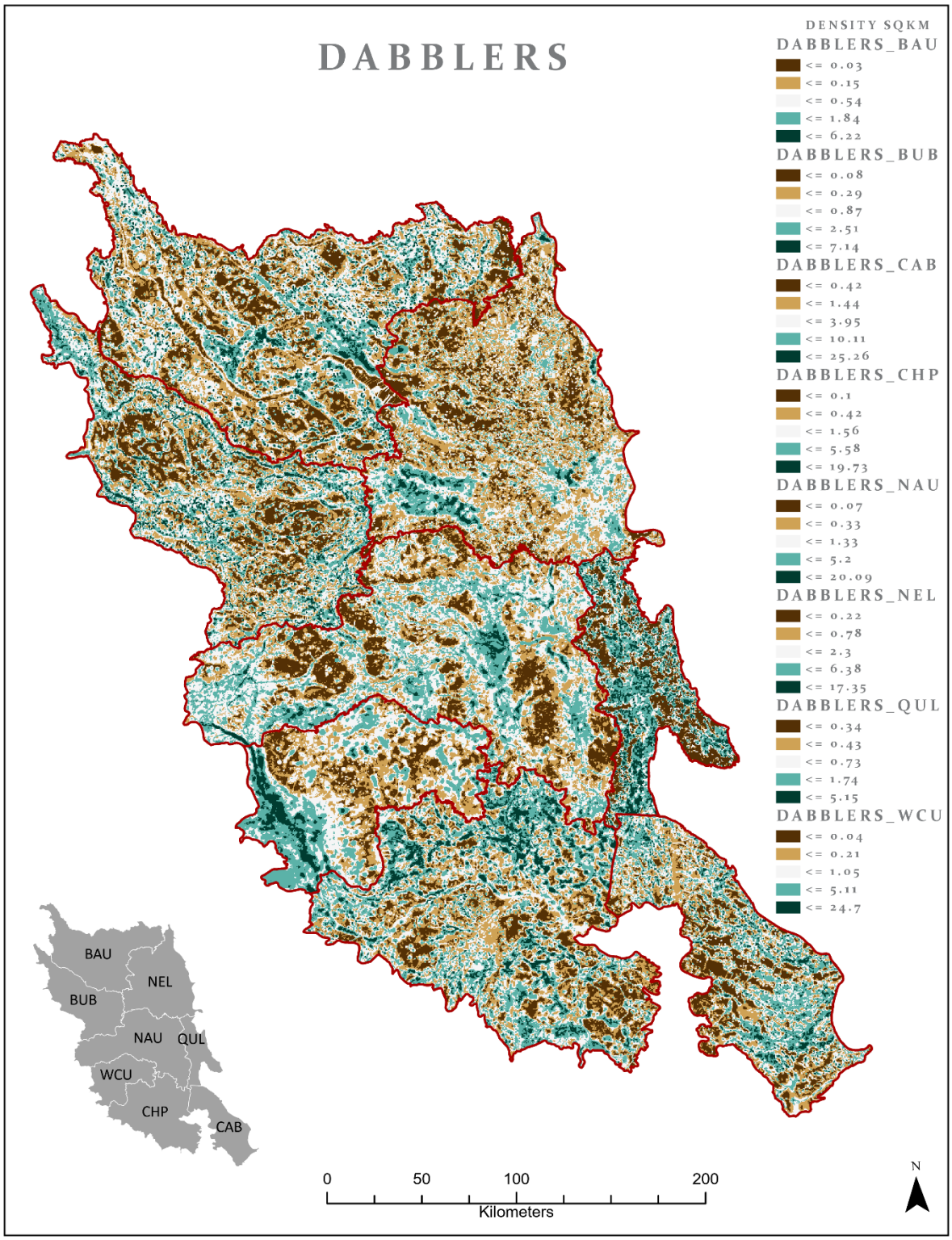


Figure A1-12. Population density per square kilometre forecast for dabblers. Density classes have been mapped according to a geometric classification for each ecosection.

DIVERS

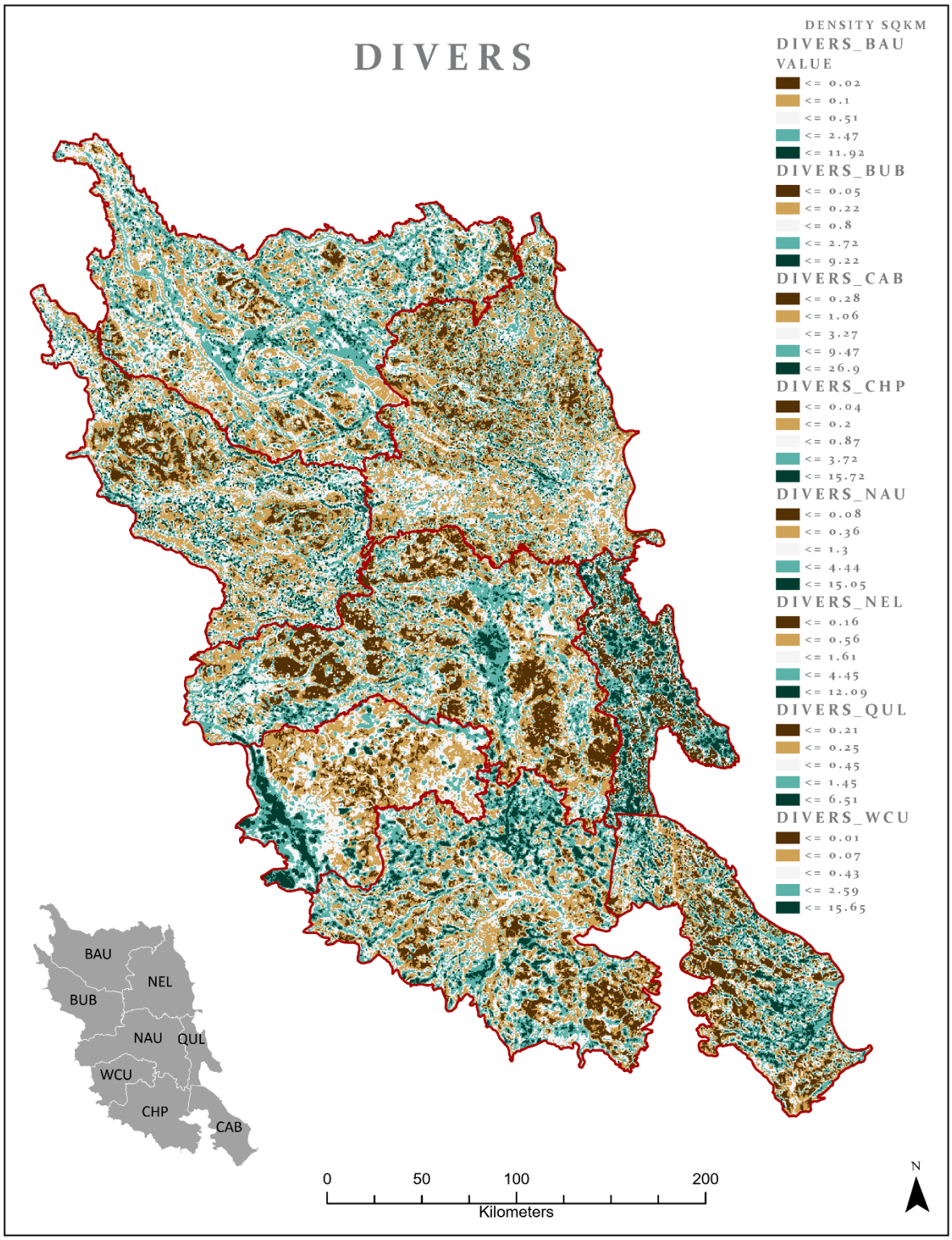


Figure A1-13. Population density per square kilometre forecast for divers. Density classes have been mapped according to a geometric classification for each ecosection.

ALL SPECIES

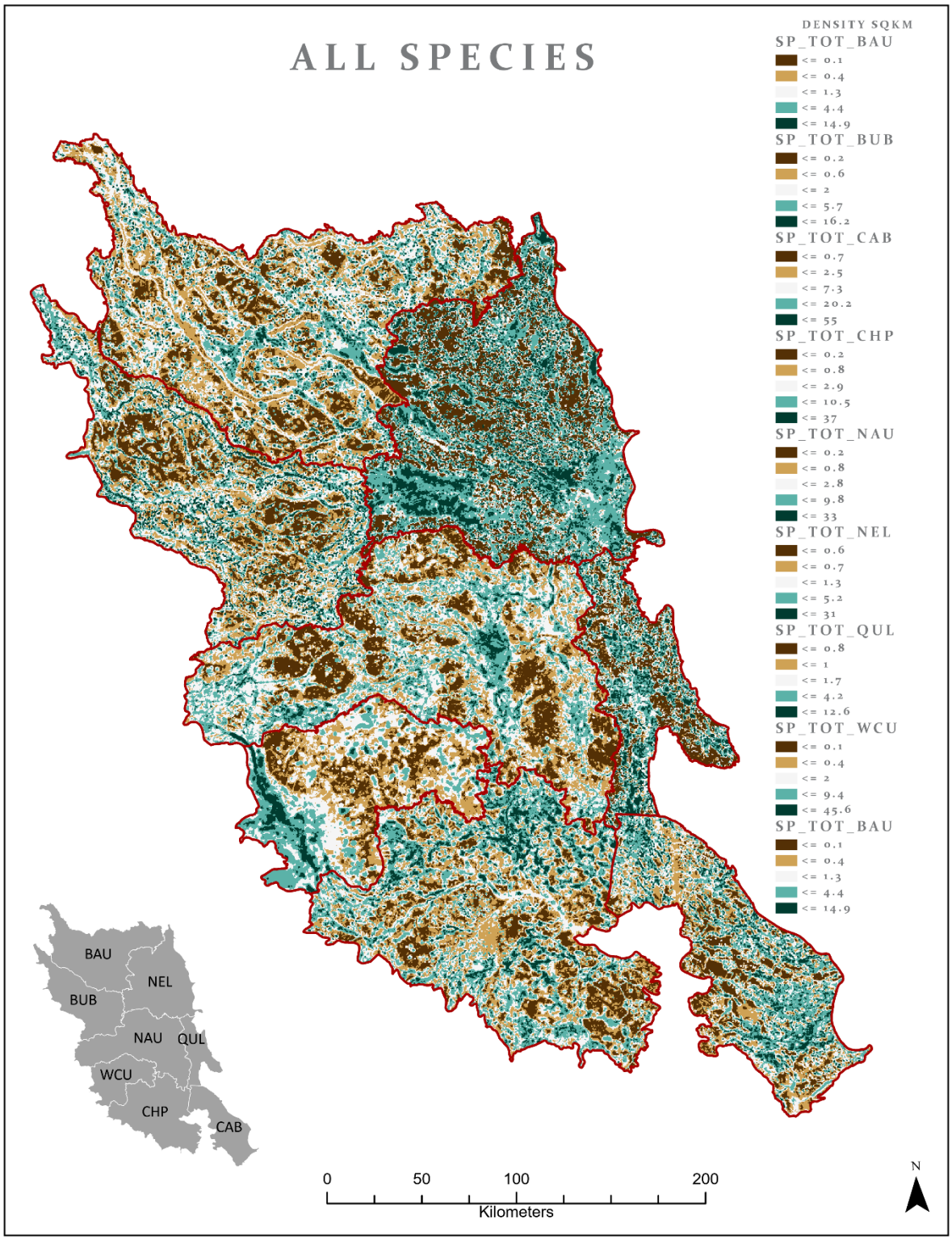


Figure A1-14. Population density per square kilometre forecast for all species. Density classes have been mapped according to a geometric classification for each ecosection.

SPECIES RICHNESS

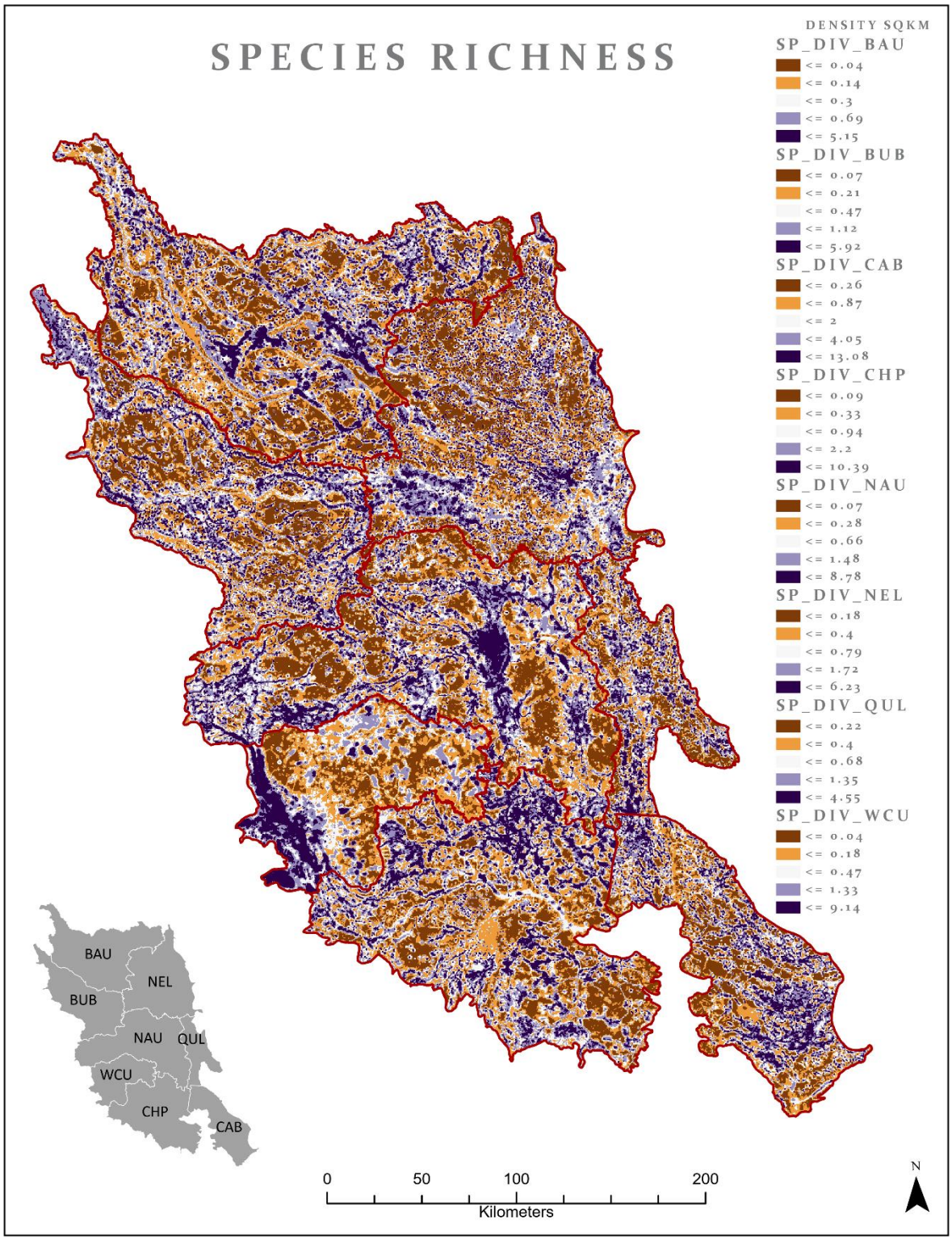


Figure A1-15. Forecast of species richness or number of unique species per square kilometre. Species richness classes have been mapped according to a quantile classification scheme for each ecosection.

APPENDIX 2 – Dataset Evaluation

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

Table A2-1. Predictor variables and datasets evaluated for model input.

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
maxT	Day of survey maximum temperature	1 km	Day	Natural Resources Canada	Determine Julian day of survey dates and Extract Value by Point	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
minT	Day of survey minimum temperature	1 km	Day	Natural Resources Canada	Determine Julian day of survey dates and Extract Value by Point	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
pcp	Day of survey total precipitation	1 km	Day	Natural Resources Canada	Determine Julian day of survey dates and Extract Value by Point	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
maxT_max	Maximum of maximum temperature of week preceding survey	1 km	Week	Natural Resources Canada	Derived from daily valuesCalculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
maxT_med	Median of maximum temperature of week preceding survey	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
maxT_mean	Mean of maximum temperature week preceding survey of	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
maxT_var	Variance of maximum temperature during week preceding survey	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
minT_max	Minimum of minimum temperature of week preceding survey	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
minT_med	Median of minimum temperature of week preceding survey	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
minT_mean	Mean of minimum temperature week preceding survey of	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
minT_var	Variance of minimum temperature during week preceding survey	1 km	Week	Natural Resources Canada	Calculated weekly statistical summary	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
pcp_tot	Total precipitation during the week	1 km	Week	Natural Resources Canada	Sum of daily precipitation during week preceding	Daily and derived weekly values are useful measures as preliminary trial results indicated their variable importance values however their application in modeling predicted distributions is minimal. Further explanatory modeling studies should investigate their influence.
ffd	Total frost free days preceding survey date	Scale-free (400m)	Day	ClimateBC	Derived from ClimateBC values indicating the beginning of the frost-free period	Number of frost-free days preceding the survey is a relevant measure of in-situ habitat conditions and preliminary results indicated significant variable important measure however their relevance in predicting distributions is minimal. Note there is a high degree of correlation and multicollinearity between 'ffd' and average spring temperatures and growing degree days)
pas	Precipitation as snow	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	Suffix indicates temporal measure http://www.climatewna.com/help/ClimateBC/Help.htm
ppt	Precipitation	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	Was not included as CMD incorporates precipiations
dd5	Growing degree days over 5°C	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	Indicative of primary productivity potential

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
dd18	Heating degree days over 18°C	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	
tave	Average temperature	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	Average temperature - influences physiological processes
cmd	Hargreaves climatic moisture deficit	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	Index of reference evaporation minus precipitation
shm	Summer heat moisture index	Scale-free (400m)	Variable - annual, seasonal, monthly	ClimateBC	Derived from coordinate location and elevation of the 400m interval along strip transect	See Bunnell et al (2010) recommended formulating a drying index using 'pas' and 'shm'
bec_zn	Biogeoclimatic Zone	Variable 1:20,000 to 1:600,000	2018	DataBC		Alternative classification to ecosection however misbalanced classes
bec_sz	Biogeoclimatic Subzone	1:20,000	2018	DataBC		Widely mis-balanced classes and survey transect is not fully representative
da	Drainage areas	1,250,000	#N/A	DataBC		
sda	Drainage sub-areas	1:50,000	#N/A	DataBC		Incorporate for modeling of hydrology
ssda	Sub-sub drainage areas	1:50,000	#N/A	DataBC		Incorporate for modeling of hydrology
watershed	Watershed ID	1:50,000	#N/A	DatabBC		Incorporate for modeling of hydrology
snow_basin	Snow Basins	1:50,000	#N/A	DataBC		Snow basins could be incorporated with drainage basins but note non-corresponding boundaries
eco	Ecosection (class of Ecoregions of BC)	1:20,000	2006	DataBC		Subset survey dataset for ecosection-specific models

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
elev	Elevation	1:20,000	#N/A	TRIM		Include if direct measures of elevation-influenced variables are not incorporated or if perform poorly
slp	Slope	1:20,000	#N/A	TRIM	Derived from DEM	Potential for wetland formation
asp	Aspect	1:20,000	#N/A	TRIM	Derived from DEM	Influences sun exposure and primary productivity
dem_mean	Average elevation	1:20,000	#N/A	TRIM		Average within the focal neighbourhood
dem_range	Range of elevation	1:20,000	#N/A	TRIM		Topographic roughness index
pa	Protected Areas	Variable	2018	CWS		Not predictive, but useful measure to determine whether areas of high density are protected
alr	Agricultural Land Reserve	1:20,000	2019	DataBC		Highly correlated with cropland land cover
survey_day	Julian day of the year	#N/A	#N/A	survey data		Used ot determine Frost-free days before survey
survey_order	Sequential order of survey	#N/A	#N/A	survey data		Ordering of the survey day within the survey year - indicates whether early or late
fwa_ow	Open water - Lakes, Manmade Water bodies, Rivers (excl wetlands)		Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
fwa	Total lake area	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
fwa_1(to 8)	Total lake area by size class	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
fwa_cnt_1(to 8)	Feature count by size class	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	Total count of wetlands by size class
fwa_10ha_(more/less)	Grouped lake class of lakes either greater or less than 10 ha	1:20,000			Total area within 1.2 km circular radius of centroid point along 400m interval of transect	

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
fwa_r	Total river area	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
fwa_w	Total wetland area	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
shore_1(to 8)	Total shoreline length by size class	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
shrcx_10plus(less)	Total shoreline length of lakes and manmade waterbodies	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
shore_lxr	Total shoreline length of open water (LRX)	1:20,000	Variable	Freshwater Atlas, DataBC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
str_s(m,l)	Stream lengths combined 'str_s', 'str_m' 'str_l' - grouped by order	1:20,000	Variable	Freshwater Atlas, DataBC	Total length within 1.2 km circular radius of centroid point along 400m interval of transect	Note str_l is represented by areal representation of rivers
urban	Land cover - urban	30m	2010	Centre for Economic Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	Highly correlated with paved roads from digital atlas
shrubland	Land cover - shrubland	30m	2010	Centre for Economic Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
mixed_forest	Land cover - mixed forest	30m	2010	Centre for Economic Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation	Notes & Recommendations
broadleaf	Land cover - broadleaf (deciduous)	30m	2010	Centre for Economic Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
needleleaf	Land cover - needleleaf (coniferous)	30m	2010	Centre for Economic Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	
mpb	Mountain Pine Beetle	1:50,000	1999-2015	Ministry of Forests, Lands, Natural Resource Operations	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	Not included due to extensive range - and inability to interpret the results or expected results
ctblk	Cutblocks	1:20,000	Variable within 40 years of logging	Ministry of Forests, Lands, Natural Resource Operations	Total area within 1.2 km circular radius of centroid point along 400m interval of transect	High correlation with needleleaf land cover
dra_u	Unpaved Roads	1:20,000		DRA - DataBC	Total length within 1.2 km circular radius of centroid point along 400m interval of transect	Highly correlated with needleleaf forest (logging)
dra_p	Paved Roads	1:20,001		DRA - DataBC	Total length within 1.2 km circular radius of centroid point along 400m interval of transect	Highly correlated with urban land cover

APPENDIX 3—Reclassification

Table A6-1. Centre for Economic Cooperation Land Cover (2010) Reclassification crosswalk table.

Original code	Original Description	New code	New Classification
1	Temperate or sub-polar needleleaf forest	1	needleleaf
2	Sub-polar taiga needleleaf forest	1	needleleaf
5	Temperate or sub-polar broadleaf deciduousorest	2	broadleaf
6	Mixed forest	3	mixed_forest
8	Temperatre or sub-polar shrubland	4	shrubland
10	Temperate or sub-polar grassland	4	shrubland
11	Sub-polar or polar shrubland-lichen-moss	5	lichenmoss
12	Sub-poolar or polar grassland-lichen-moss	5	lichenmoss
14	Wetland	6	wetland
15	Cropland	7	cropland
16	Barren Lands	8	barren
17	Urban and Built-up	9	urban
18	Water	10	water
19	Snow and Ice	11	snowice

APPENDIX 4 – Python Script Toolbox

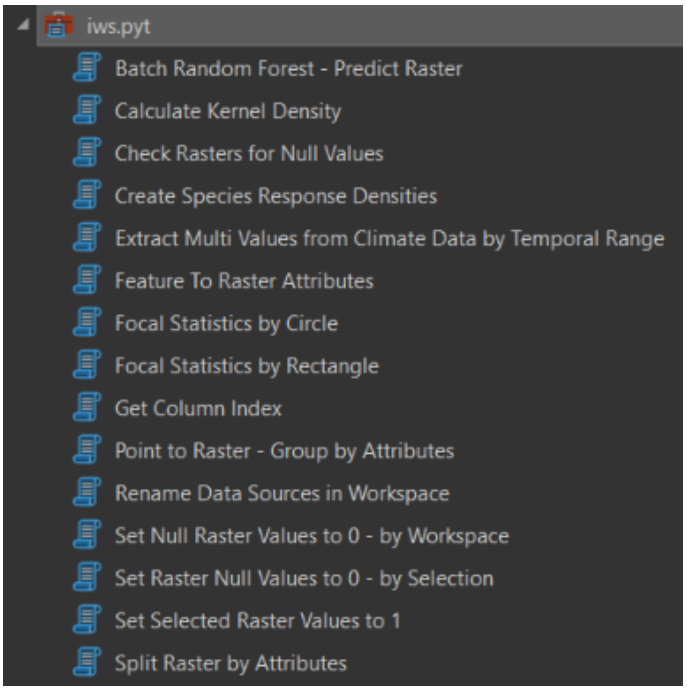


Figure A4-1. Python Script Toolbox as displayed in ArcGis Pro.

The code for the Python script toolbox is provided below.

```
# -*- 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,
                      setRasterNullValuesToOWorkspace,
                      renameDataInWorkspace,
                      kernelDensityCalculations,
                      splitRasterByAttributes,
                      focalStatsByCircle,
                      focalStatsByRectangle,
                      setRasterValuesTo1,
                      pointToRasterGroupByAttributes,
                      getColumnIndex,
                      extractClimateValuesForRange,
                      checkRasterForNullValues,
                      batchRandomForest,
                      createPredictedDensitySurfaces]
```

```
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",
            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
        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 setRasterNullValuesToOWorkspace(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"
        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 setRasterNullValuesToO(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Set Raster Null Values to O - by Selection"
        self.description = "Sets all null raster values to O 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))
            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(
            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(
            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):
        """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"),
                                 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",
            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",
            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

    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", "taveO4"]

        # 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])
            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 5 – 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

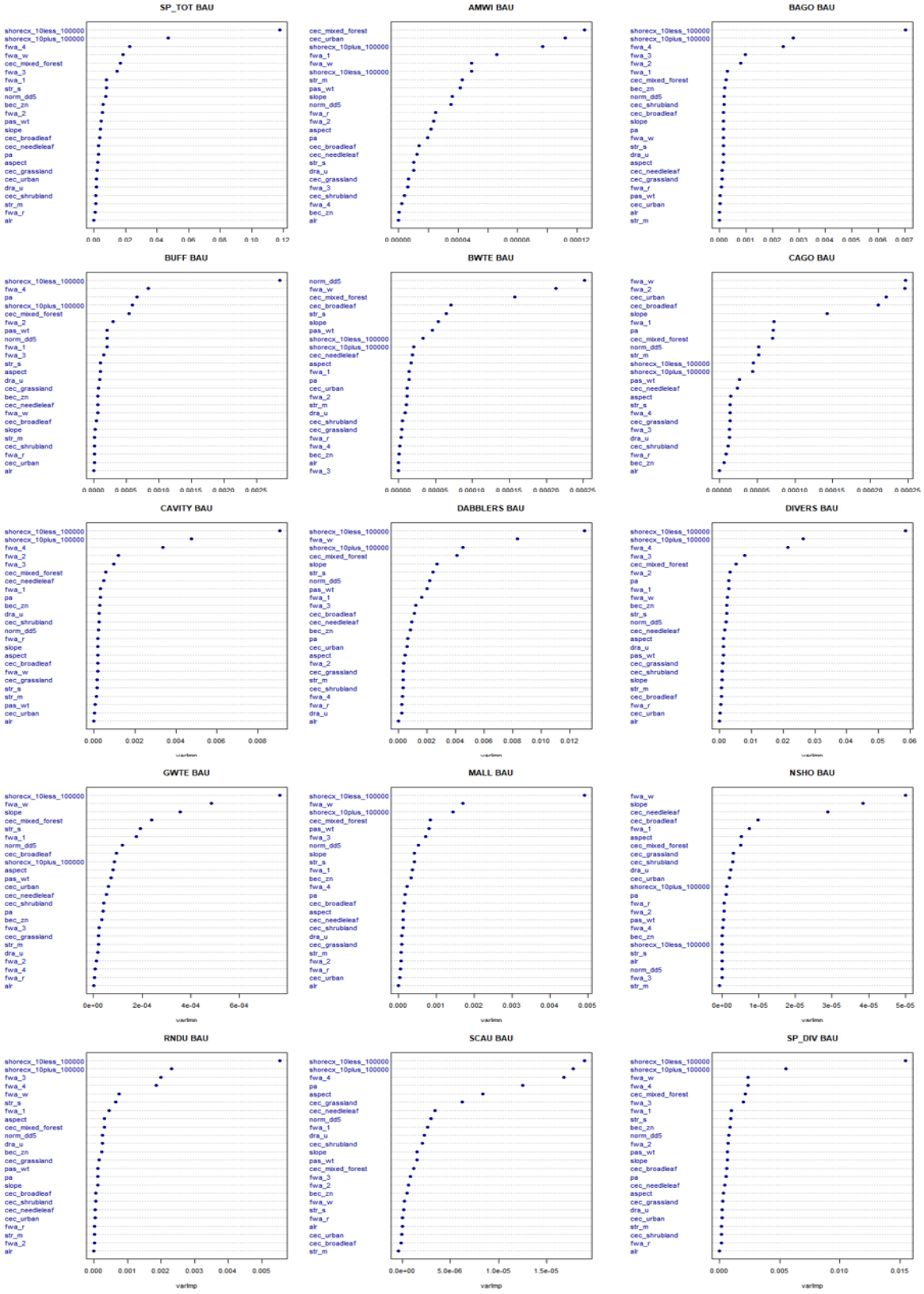
```
# For data pre-processing steps please refer to the Github repository and the Jupyter Notebook

# 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);
    }

    NA# 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)
  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)
```

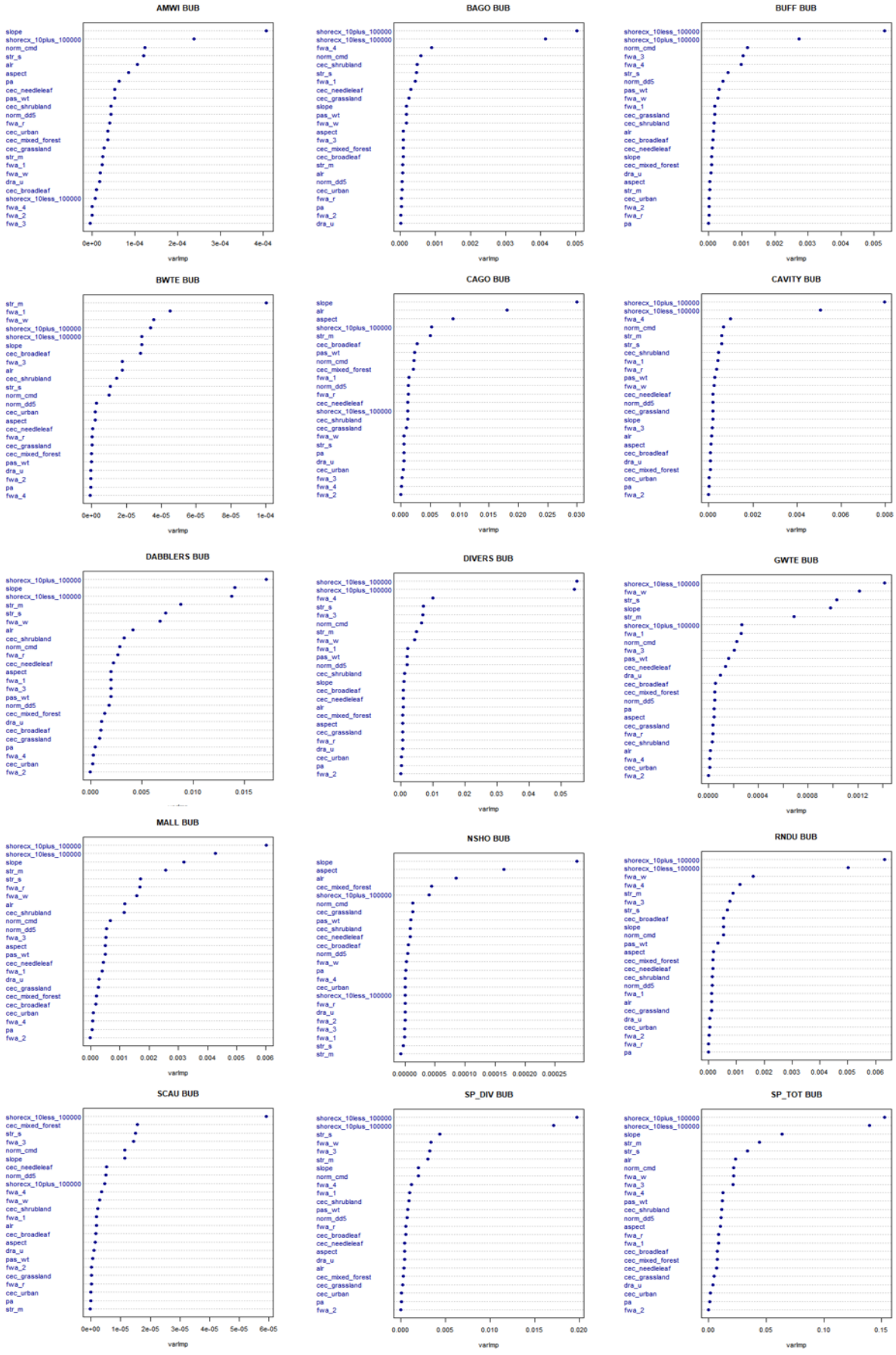
```
    v <- as.vector(df_vi$vi)

    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")
```

APPENDIX 6 – Variable Importance Plots Dot Charts

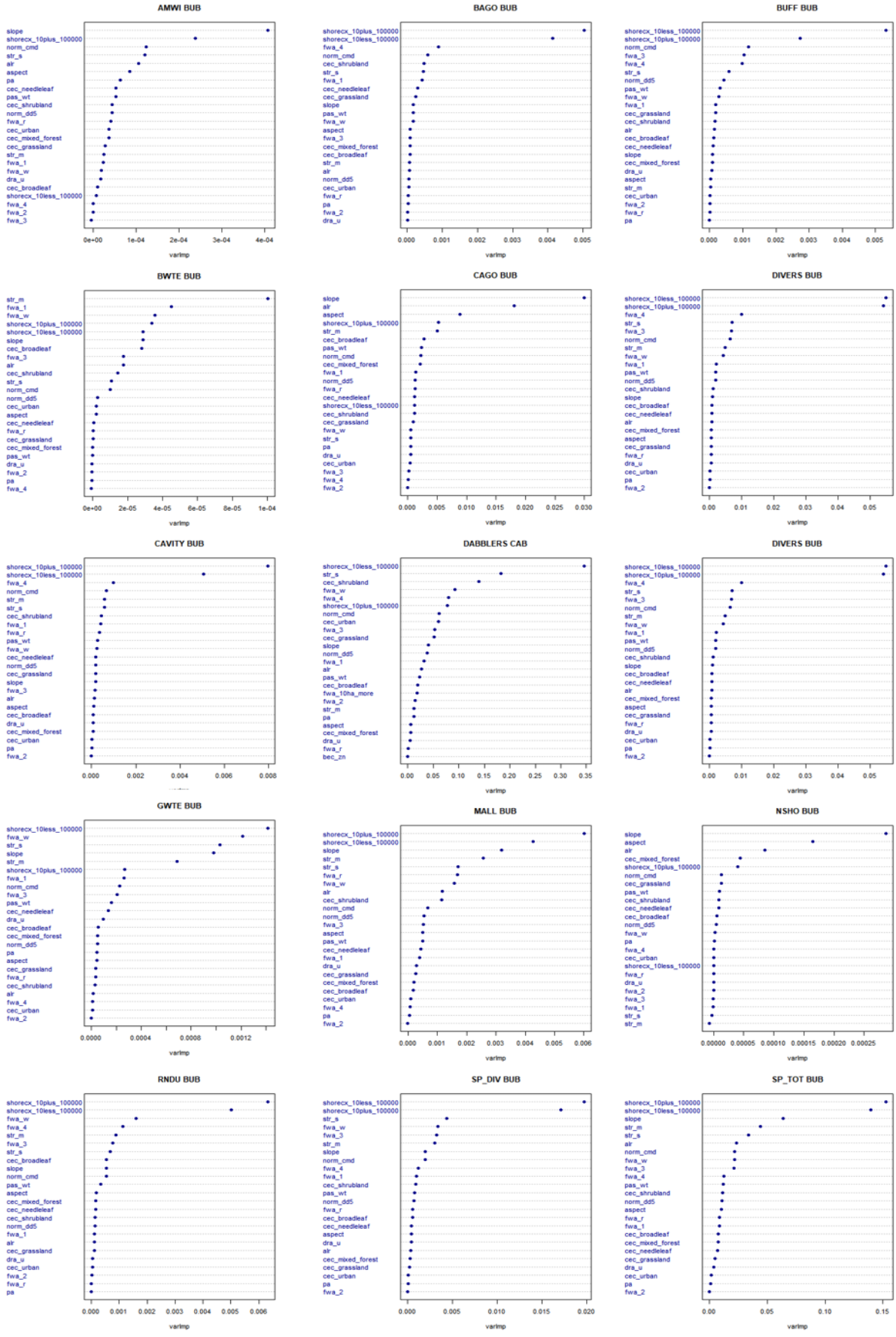
Babine Upland



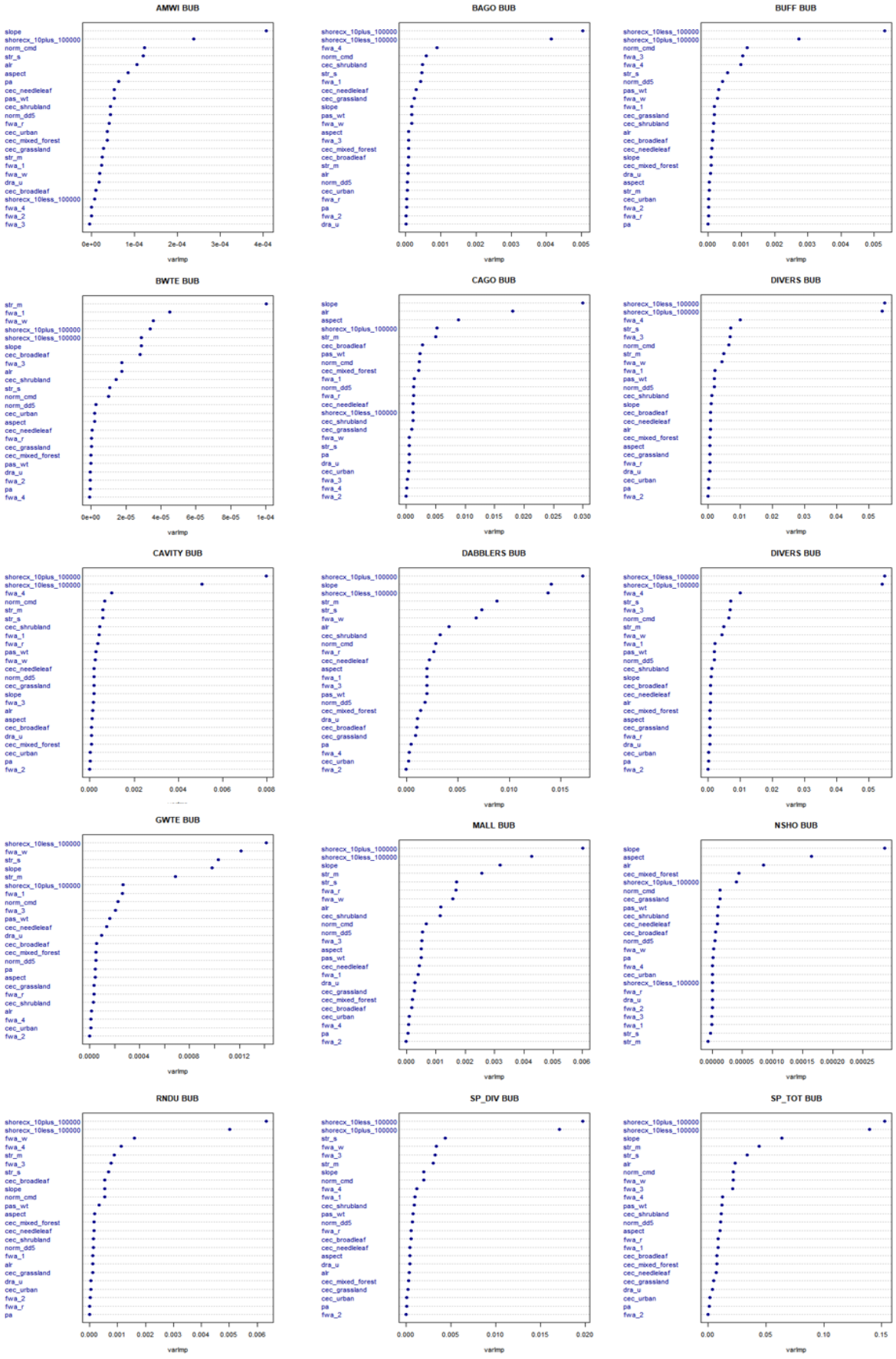
Bulkley Basin



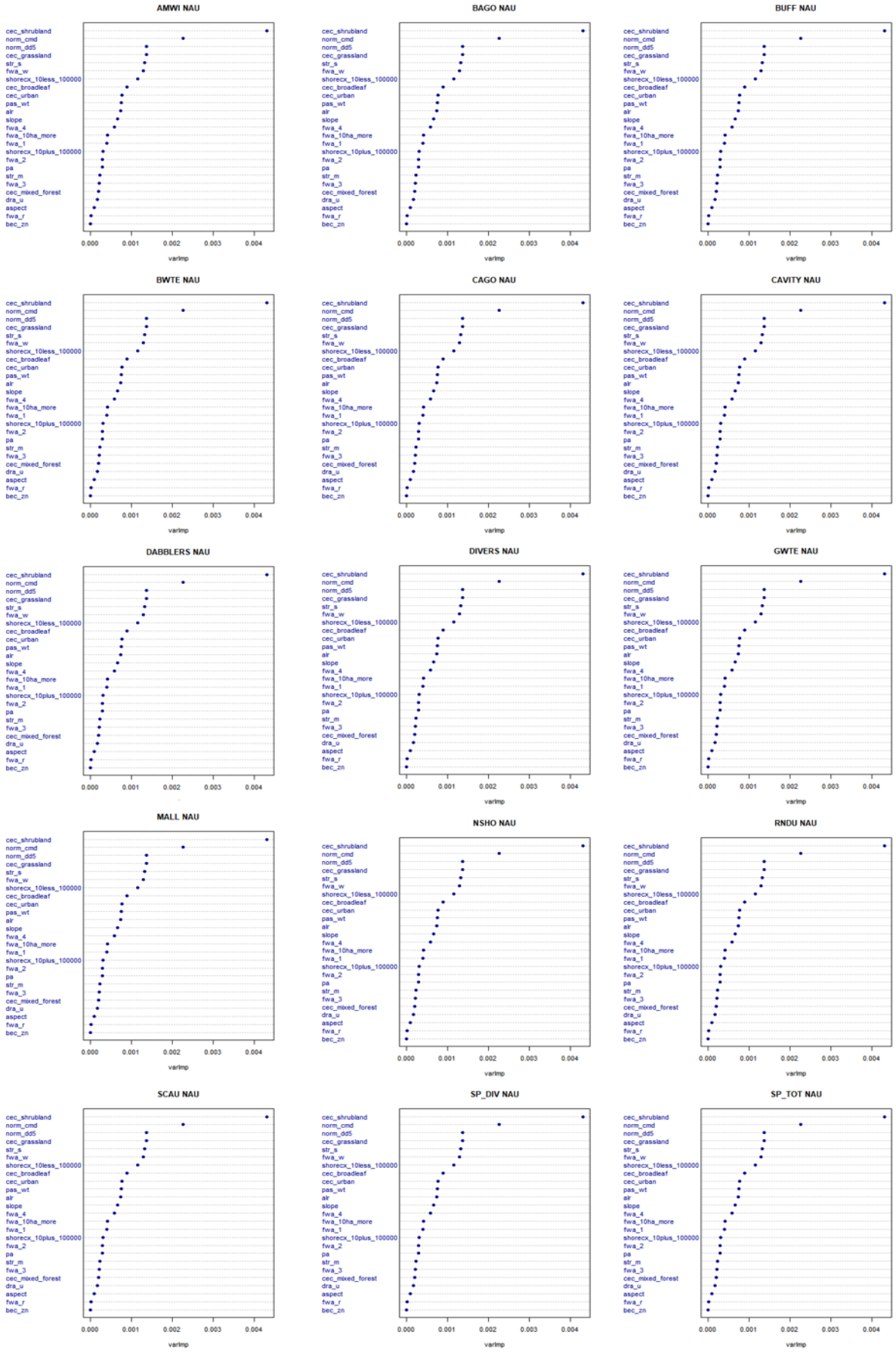
Cariboo Basin



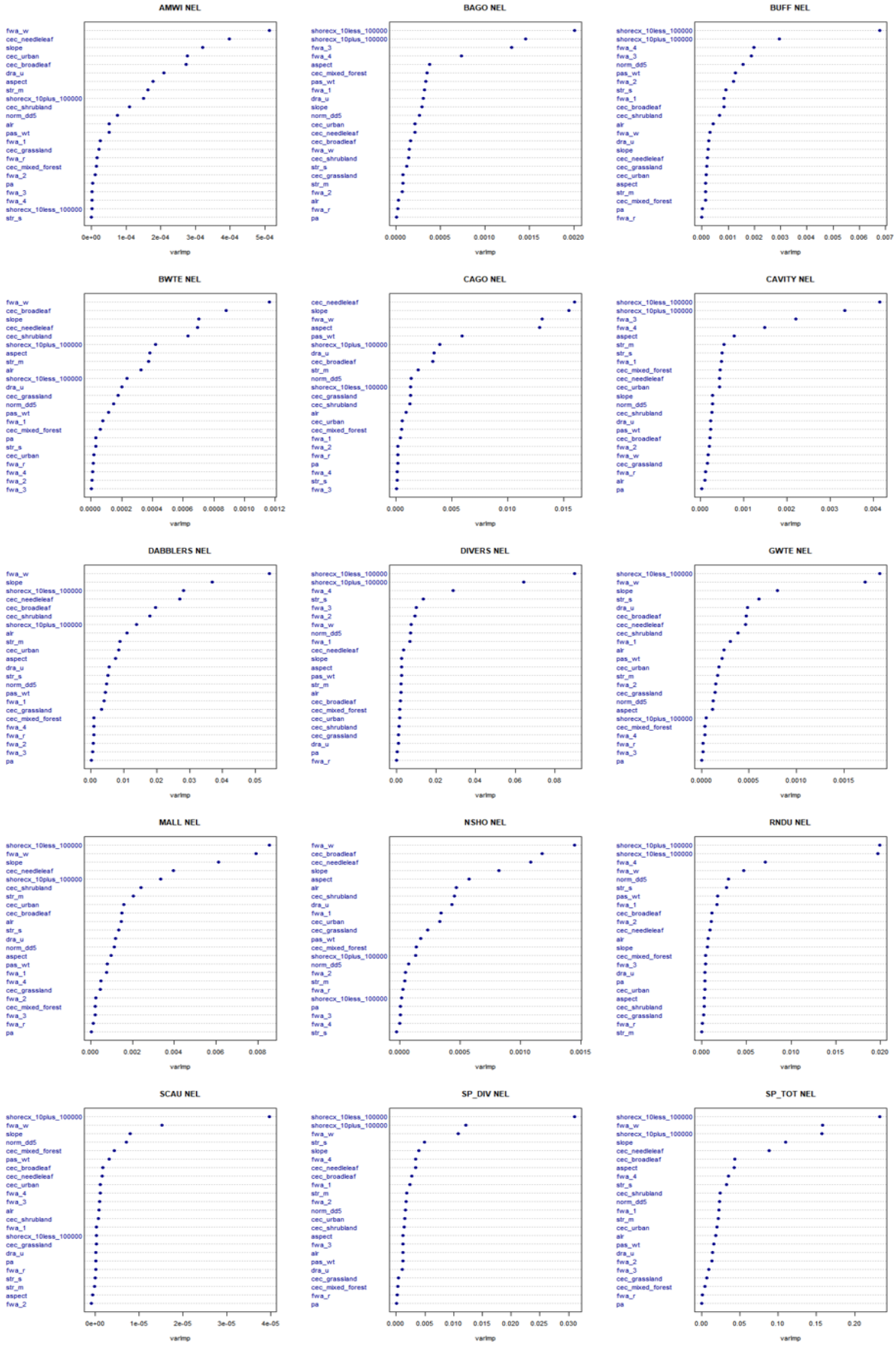
Chilcotin Plateau



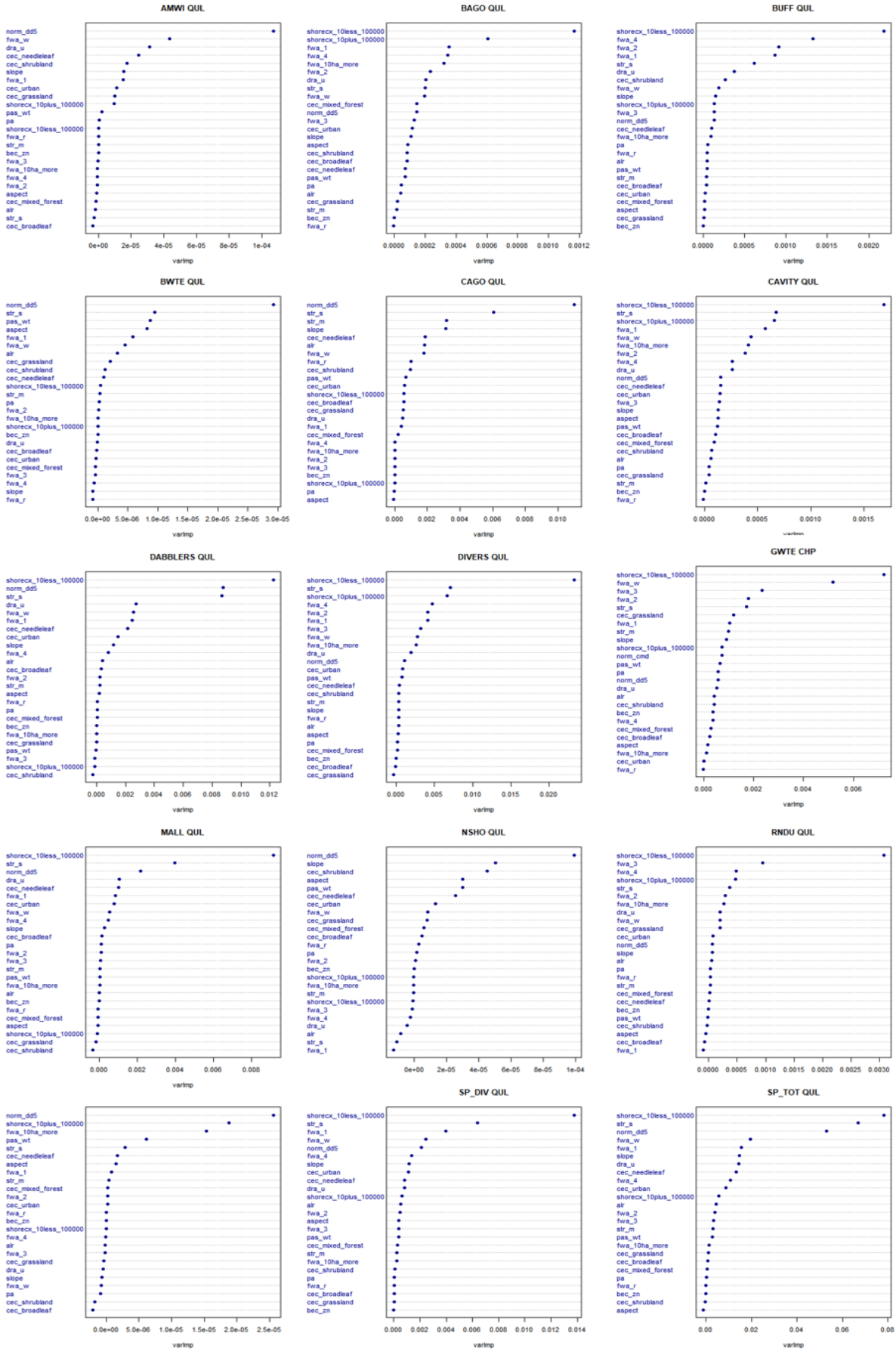
Nazko Upland



Nechako Lowland



Quesnel Lowland



Western Chilcotin Upland