Appendix I: simulated landscapes

Required packages

In the first step, we load all of the required packages. Note that grainchanger (the package developed for this paper) can be installed using:

```
devtools::install_github("laurajanegraham/grainchanger")
```

```
# load required libraries
library(captioner)
library(knitr)
library(raster)
library(grainchanger)
library(broom)
library(landscapetools)
library(NLMR)
library(tidyverse)
library(patchwork)
library(cowplot)

# set up plotting options
theme_set(theme_bw(base_size = 7) + theme(strip.background = element_blank(),
                                          panel.grid.major = element_blank(),
                                          panel.grid.minor = element_blank()))

# required for captioning and numbering tables and figures
tabs <- captioner(prefix = "Table AI.")
figs <- captioner(prefix = "Figure AI.")
```

Moving window approach

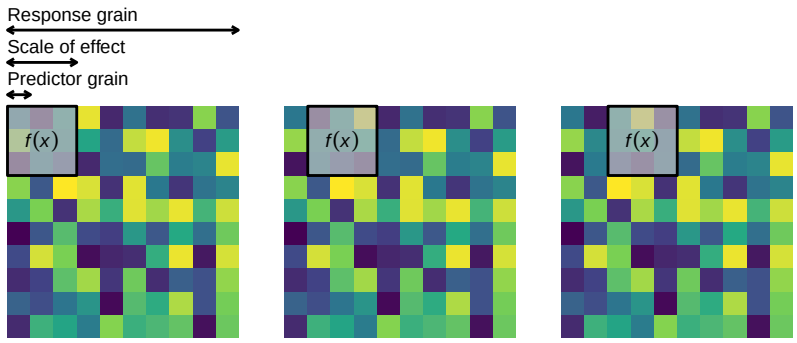


Figure AI. 1: Graphical representation of the moving window method. When the moving window is defined, three aspects of scale need to be considered. Predictor grain is the characteristic spatial scale of the predictor

variable, i.e. the resolution of the environmental data; scale of effect determines the appropriate scale of the relationship between predictor and response, for example an ecological neighbourhood; response grain is the grain of the unit into which you are predicting, i.e. the resolution of the response variable.

There are three steps involved in our data-aggregation approach: 1) define the appropriate scales for the ecological process; 2) define the appropriate measure of environmental heterogeneity and calculate using a moving window; 3) calculate the mean of the moving window-based measure at the grain of the response (Figure AI. 1).

Simulated landscapes

1. Simulate study landscapes each with 100 coarse grid cells (each of 65 x 65 fine grid cells) with varying combinations of spatial autocorrelation using `nlm_fbm` from the `NLMR` package [@Sciaini2018]. This is the fractal Brownian motion method where `fract_dim` controls the level of spatial autocorrelation in the landscape. A value close to zero is a rough landscape (random), and a value of one is a smooth landscape (spatially autocorrelated). We used 5 combinations of spatial autocorrelation:

- No spatial autocorrelation: `fractal_dim` = 0.1 for all landscapes
- Low, varied spatial autocorrelation: `fractal_dim` in range 0.1–0.5
- Varied spatial autocorrelation: `fractal_dim` in range 0.1–1
- High, varied spatial autocorrelation: `fractal_dim` in range 0.5–1
- High spatial autocorrelation: `fractal_dim` = 1 for all landscapes

2. Create categorical landscapes with 5 land cover classes by using `util_classify` with random weights for each class.

3. Use `winmove` from the `grainchanger` package to calculate the moving window measures (MWM) for each coarse grid cell. For continuous landscapes, this is variance; for categorical landscapes, this is Shannon evenness. We do this for 5 window sizes:

- 3 x 3 pixels (0.2% of coarse grid cell)
- 9 x 9 pixels (1.9% of coarse grid cell)
- 15 x 15 pixels (5.3% of coarse grid cell)
- 35 x 35 pixels (29% of coarse grid cell)
- 55 x 55 pixels (71.6% of coarse grid cell)

4. Calculate the corresponding measures for each coarse grid cell using standard (non-moving window) methods (LSM).

We created 100 replicate study landscapes using the Iridis HPC at University of Southampton. The below code is for one replicate:

```
library(raster)
library(NLMR)
library(landscapetools)
library(grainchanger)
library(tidyverse)
library(furrr)

plan(multiprocess)

# 1. Create the landscapes and calculate LS and MW measures
# Set up spatial autocorrelation parameters

strt <- Sys.time()
res <- bind_rows(
  tibble(sa_scenario = "No spatial autocorrelation", sa_values = rep(0.1, 100)),
```

```
  tibble(sa_scenario = "Low, varied spatial autocorrelation", sa_values = seq(0.1, 0.5, length.out = 10
  tibble(sa_scenario = "Varied spatial autocorrelation", sa_values = seq(0.1, 1, length.out = 100)),
  tibble(sa_scenario = "High, varied spatial autocorrelation", sa_values = seq(0.5, 1, length.out = 100
  tibble(sa_scenario = "High spatial autocorrelation", sa_values = rep(1, 100))
) %>%
  # create the continuous and categorical landscapes
  mutate(cont_ls = future_map(sa_values, function(x) nlm_fbm(ncol = 400, nrow = 400, resolution = 25, f
         # the categorical map has a random proportion between landscapes for each landscape
         cat_wt = future_map(sa_values, function(x) diff(c(0, sort(runif(4)), 1))),
         cat_ls = future_map2(cont_ls, cat_wt, function(x, wt) util_classify(x, weighting = wt)),
         # calculate LSM
         cont_lsm = future_map_dbl(cont_ls, function(x) x %>% raster::values() %>% var),
         cat_lsm = future_map_dbl(cat_ls, function(x) diversity(x, lc_class = 0:4))) %>%
  crossing(window = c(500, 1000, 1500, 3500)) %>%
  mutate(cont_ls_pad = future_map2(cont_ls, window, create_torus),
         cat_ls_pad = future_map2(cat_ls, window, create_torus),
         # calculate MWM
         cont_mwm = future_map2_dbl(cont_ls_pad, window, function(ls, w) winmove(ls, d = w, type = "rec
         cat_mwm = future_map2_dbl(cat_ls_pad, window, function(ls, w) winmove(ls, d = w, type = "recta
  ) %>% select(-cont_ls, -cat_ls, -cont_ls_pad, -cat_ls_pad)

runtime = difftime(Sys.time(),  strt)
out = list(res = res, runtime = runtime)
save(out, file = tempfile(tmpdir = ".", fileext = ".Rda"))
```

We use the simulated landscapes to explore two questions:

1. What is the correlation between the moving window and standard aggregations at different levels of spatial autocorrelation and moving window sizes?
2. Can we successfully identify the scale of effect?

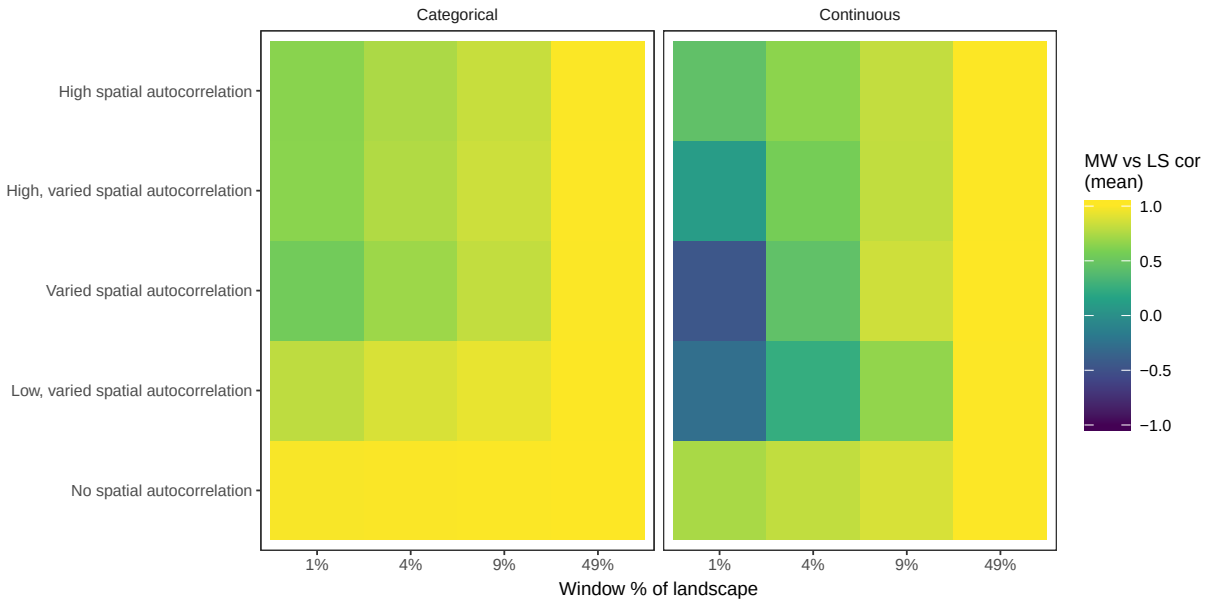
Question 1: correlation between MW and LS

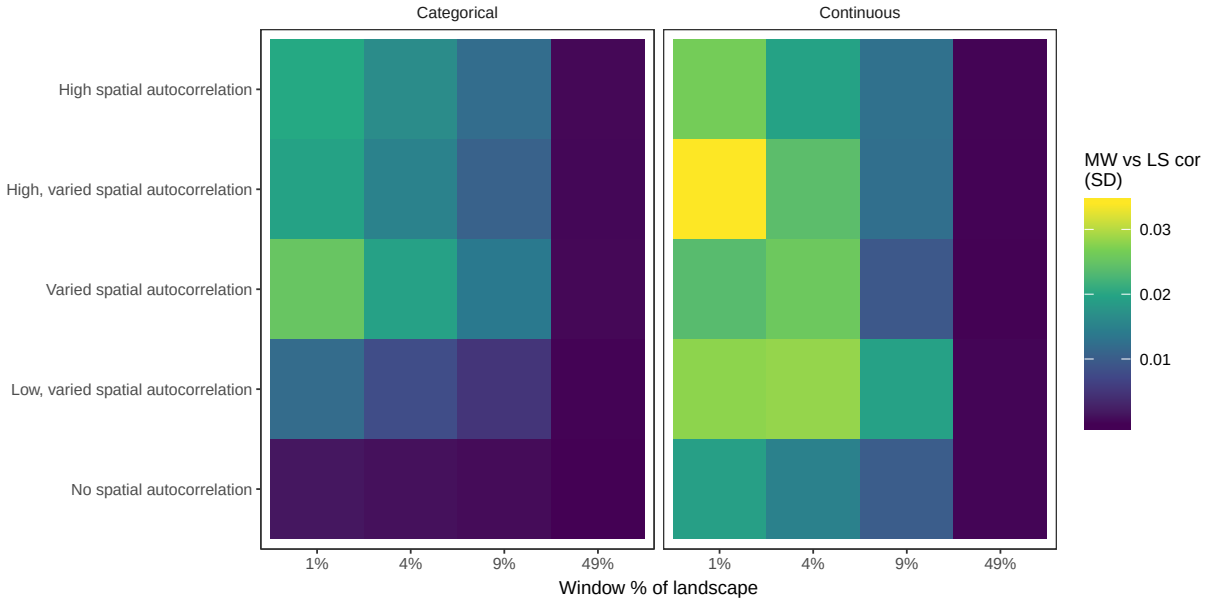
Here we calculate the Spearman's rank correlation between LSM and MWM for each replicate at each level of spatial autocorrelation and moving window size. The plot below shows the mean and standard deviation of these correlations (across 100 replicates).

```
load("results/sims_all_replicates.Rda")

df <- df %>%
  mutate(sa_scenario = factor(sa_scenario,
                              levels = c("No spatial autocorrelation",
                                         "Low, varied spatial autocorrelation",
                                         "Varied spatial autocorrelation",
                                         "High, varied spatial autocorrelation",
                                         "High spatial autocorrelation")),
         window = factor(window, labels = unique(paste0(round(((2*window + 1)^2/10000^2)*100, 1), "%"))
         rep = ceiling((rep %>% as.factor %>% as.numeric)/10)) %>%
  group_by(rep, sa_scenario, window) %>%
  mutate(ls_id = 1:n()) %>%
  ungroup()
```

```
corrs <- df %>%
  group_by(rep, sa_scenario, window) %>%
```

```
  nest() %>%
  mutate(
    cont_cor = map_dbl(data, function(x) {
    cor.test(x$cont_lsm, x$cont_mwm, method = "spearman")$estimate
  }),
  cat_cor = map_dbl(data, function(x) {
    cor.test(x$cat_lsm, x$cat_mwm, method = "spearman")$estimate}
  )) %>%
  select(-data) %>%
  group_by(sa_scenario, window) %>%
  summarise_at(.vars = vars(cont_cor, cat_cor), .funs = c("mean", "sd")) %>%
  gather(measure, value, -sa_scenario, -window) %>%
  mutate(measure = factor(measure,
                          labels = c("Categorical_mean", "Categorical_sd",
                                     "Continuous_mean", "Continuous_sd"))) %>%
  separate(measure, into = c("ls_type", "measure_type"), sep = "_")

# 1. correlations between MW and LS for diff spatial autocorrelation and window sizes
p1 <- ggplot(corrs %>% filter(measure_type == "mean"),
       aes(x = window, y = sa_scenario, fill = value)) +
  geom_tile() +
  scale_fill_viridis_c("MW vs LS cor\n(mean)", limits = c(-1, 1)) +
  labs(x = "Window % of landscape", y = "") +
  facet_wrap(~ls_type)

p2 <- ggplot(corrs %>% filter(measure_type == "sd"),
             aes(x = window, y = sa_scenario, fill = value)) +
  geom_tile() +
  scale_fill_viridis_c("MW vs LS cor\n(SD)") +
  labs(x = "Window % of landscape", y = "") +
  facet_wrap(~ls_type)


p1 / p2
```



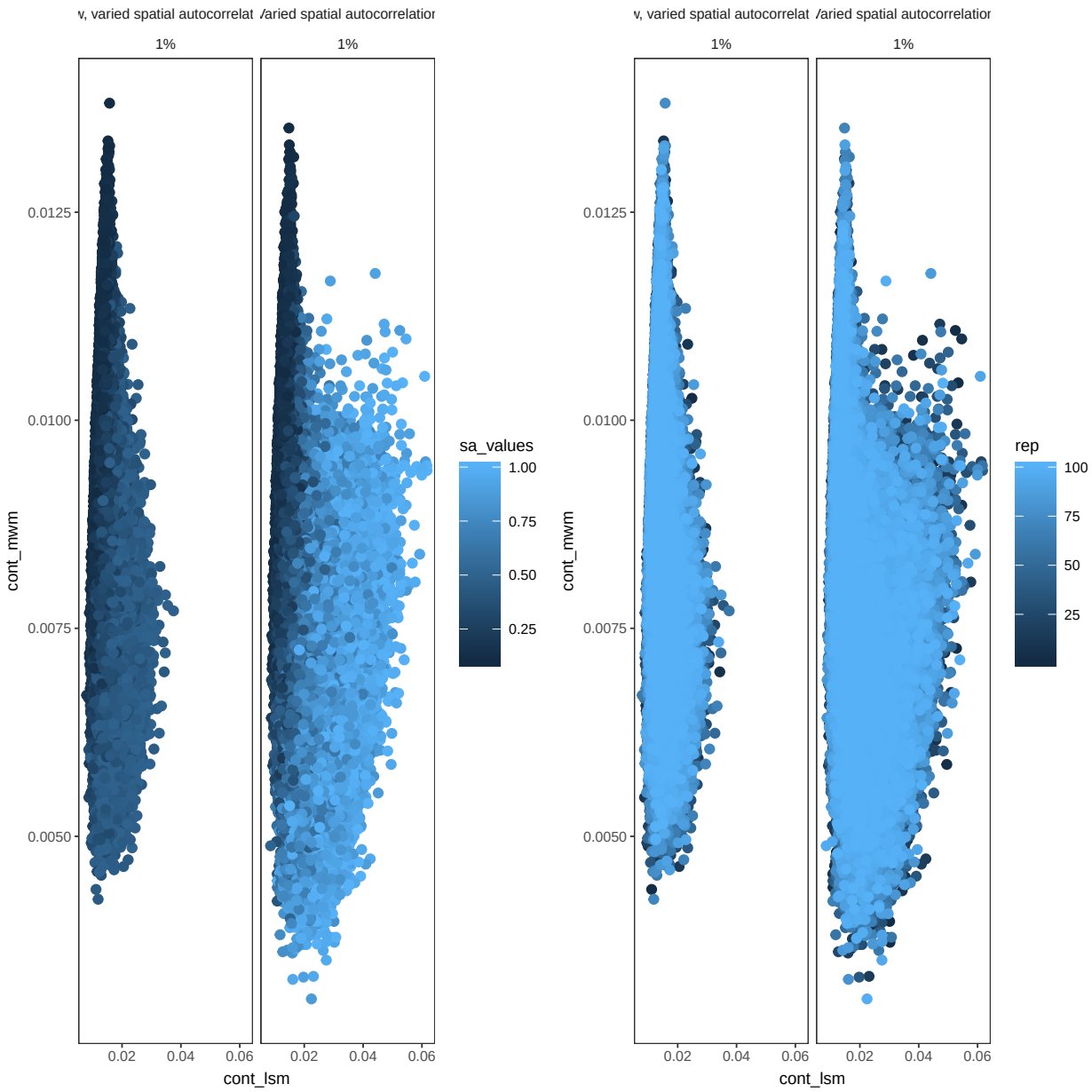


- Higher correlation between MWM and LSM in continuous landscapes
- We even get negative correlation in categorical landscapes for small moving window and variation in spatial autocorrelation.
- MW and LS are highly correlated for the two largest window sizes.
- There is more variability between the amount of correlation in replicates for continuous than categorical landscapes.

Why do we get negative correlations in some cases? The below plots show each spatial autocorrelation scenario and window size combination which results in a negative correlation between MWM and LSM measures. In the first plot, points are coloured by the spatial autocorrelation value used to generate the landscape. In the second, they are coloured by the replicate ID.

```
neg_corr <- corrs %>% filter(value < 0) %>%
  inner_join(df)
```

```
neg_corr_p1 <- ggplot(neg_corr, aes(x = cont_lsm, y = cont_mwm, colour = sa_values, group = sa_values))
  geom_point() +
  facet_wrap(~ sa_scenario + window)

neg_corr_p2 <- ggplot(neg_corr, aes(x = cont_lsm, y = cont_mwm, colour = rep, group = rep)) +
  geom_point() +
  facet_wrap(~ sa_scenario + window)

neg_corr_p1 + neg_corr_p2
```



In all cases, low values of spatial autocorrelation result in high MWM and low LSM variance; high values of spatial autocorrelation result in low MWM and high LSM. This is what is driving the negative correlation. Within spatial autocorrelation values, we still see positive correlation between replicates, these correlations are greater for lower values of spatial autocorrelation. Within replicate, this manifests as negative correlation

(leading to an overall negative correlation when taken as the mean of all replicates).

Question 2: identifying the scale-of-effect

In order to examine whether we can correctly identify the scale of effect when comparing several correlated measures calculated using different window sizes, we use our MW measures to generate an ecological response variable for every window size in each coarse grid cell in each replicate landscape and all spatial autocorrelation scenarios. We calculate this response variable as $y_w = MW_w + \epsilon$ where y_w is the response variable for a given window size w, MW_w is the moving window calculated measure for window size w and ϵ $N(0, \sigma)$. We use three levels of ϵ:

- low ϵ: first percentile of the MWM measure value within each spatial autocorrelation and window combination. This represents data with minimal noise.
- moderate ϵ: tenth percentile of the MWM measure value within each spatial autocorrelation and window combination. This represents data with a moderate amount of noise.
- high ϵ: median of the MWM measure value within each spatial autocorrelation and window combination. This represents data with a large amount of noise.

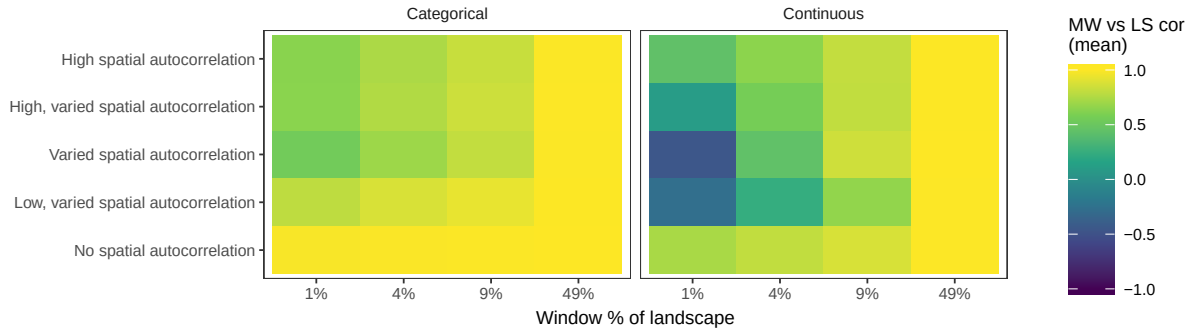
For each y_w we fit a univariate linear model with each MW_w as the covariate and use R^2 and AIC to examine whether the correct window size (and thus, scale of effect) is identified.

For the models fit for each of the y_w and MW_w combinations, we calculate the number of replicates where the best fitting model was the correct one.

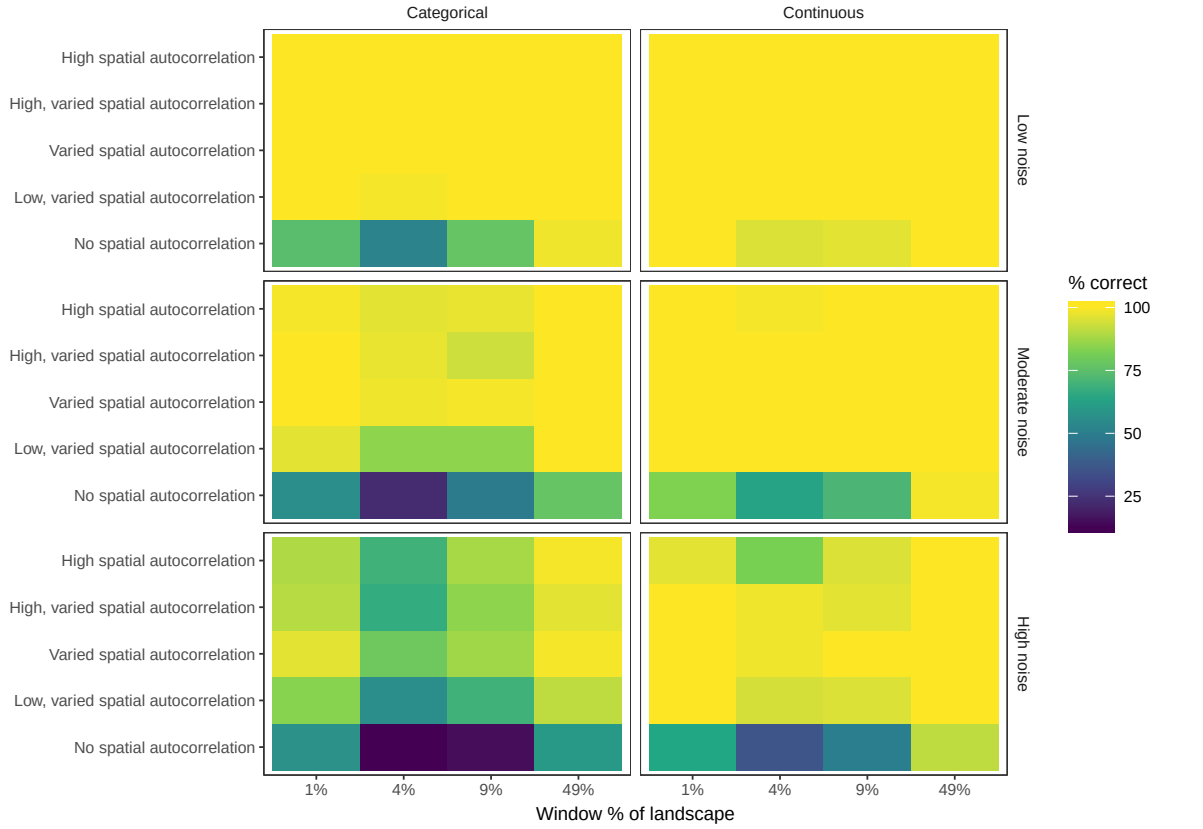
```
window_y = unique(df$window)
noise_level = tibble(noise_level = c("Low noise", "Moderate noise", "High noise"),
                     noise_value = c(0.1, 0.25, 0.5))

df_narrow <- df %>% select(rep, ls_id, sa_scenario, window, cont_mwm, cat_mwm) %>%
  gather(measure, value, -rep, -ls_id, -sa_scenario, -window)

y_vals <- df_narrow %>%
  crossing(noise_level) %>%
  group_by(sa_scenario, window, measure) %>%
  mutate(sigma = noise_value * mean(value),
         eps = map_dbl(sigma, function(x) rnorm(1, 0, x)),
         y = value + eps) %>%
  ungroup() %>%
  select(rep, sa_scenario, ls_id, window_y = window, noise_level, measure, y)

mod_df <- df_narrow %>% rename(window_x = window) %>%
  crossing(window_y) %>%
  inner_join(y_vals) %>%
  group_by(rep, sa_scenario, noise_level, measure, window_x, window_y) %>%
  nest() %>%
  mutate(mod = map(data, function(x) lm(y ~ value, data = x)),
         mod_glance = map(mod, glance),
         mod_r2 = map_dbl(mod_glance, function(x) x$r.squared),
         mod_aic = map_dbl(mod_glance, function(x) x$AIC))

correct_df <- mod_df %>%
  select(rep, sa_scenario, window_x, window_y, noise_level, measure, mod_r2, mod_aic) %>%
  gather(key, value, -rep, -sa_scenario, -window_x, -window_y, -noise_level, -measure) %>%
  mutate(value = case_when(key == "mod_r2" ~ 1 - value, TRUE ~ value),
         ls_type = factor(measure, labels = c("Categorical", "Continuous")),
         fit_measure = factor(key,
```

```
                               labels = c("AIC", expression(R^{2}))),
         noise_level = factor(noise_level,
                              levels = c("Low noise", "Moderate noise", "High noise"))) %>%
  group_by(rep, sa_scenario, window_y, noise_level, ls_type, fit_measure) %>%
  slice(which.min(value)) %>%
  mutate(pass = window_x == window_y) %>%
  group_by(sa_scenario, window_y, noise_level, ls_type, fit_measure) %>%
  summarise(pass = sum(pass)) %>%
  ungroup()

scale_plot <- ggplot(correct_df %>% filter(fit_measure == "AIC"),
       aes(x = window_y, y = sa_scenario, fill = pass)) +
  geom_tile() +
  scale_fill_viridis_c("% correct") +
  labs(x = "Window % of landscape", y = "") +
  facet_grid(noise_level ~ ls_type)

p1/scale_plot + plot_layout(heights = c(1,3)) + plot_annotation(tag_levels = "a", tag_suffix = ")")
```

a)



b)

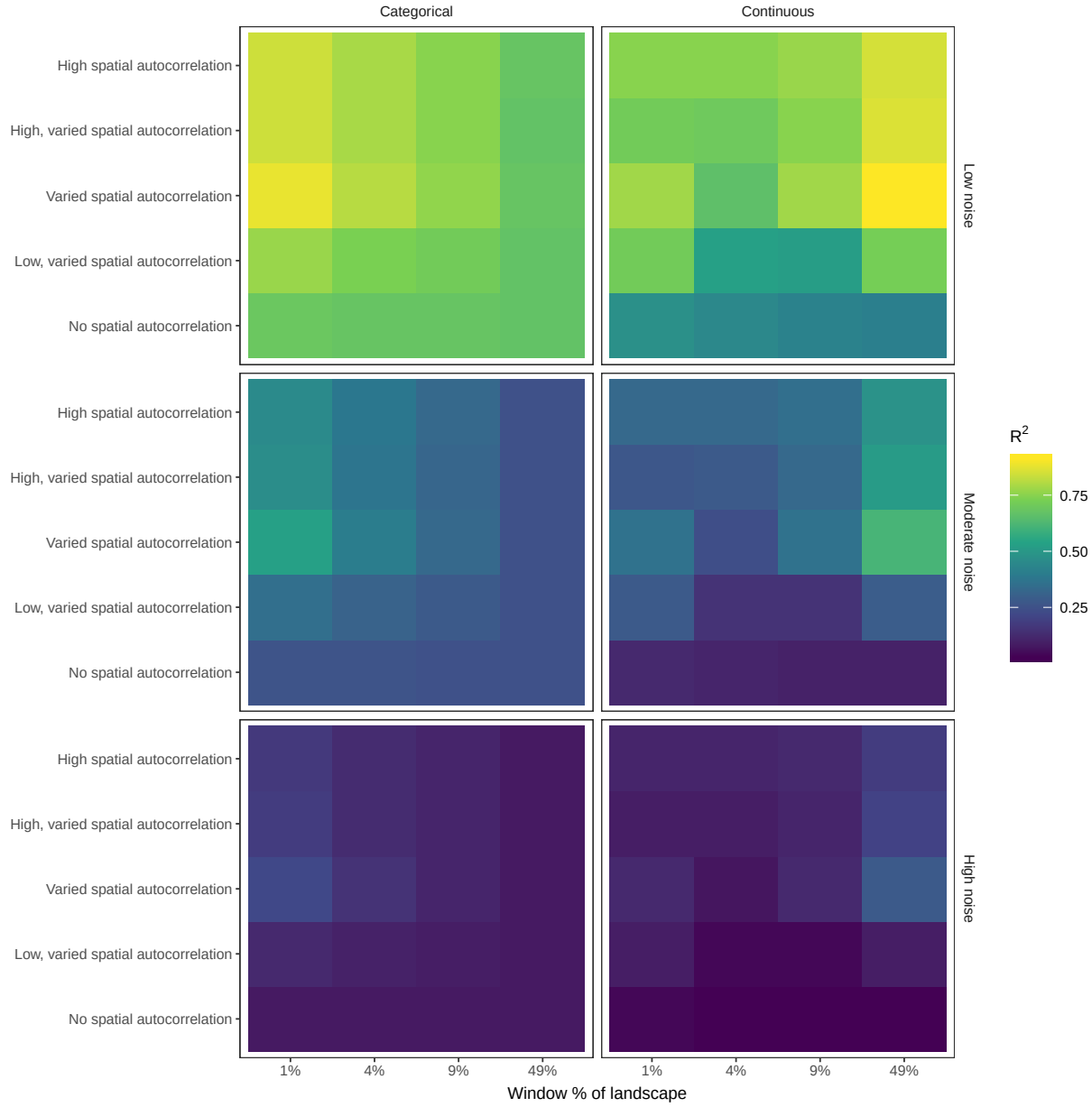


- Whether judged by R^2 or AIC, we are able to identify the correct scale of effect in low noise datasets most of the time
- In moderate and high noise datasets, we are best able to identify the correct scale of effect when it is small or large, and the data is continuous.
- Overall, we were least successful in identifying the scale of effect in the No spatial autocorrelation scenario

It's also useful to understand the general fit of the models at each spatial autocorrelation and window size combination.

```
r2_df <- mod_df %>%
  select(measure, noise_level, sa_scenario, window_x, window_y, mod_r2) %>%
  filter(window_x == window_y) %>%
  mutate(ls_type = factor(measure, labels = c("Categorical", "Continuous")),
         noise_level = factor(noise_level,
```

```
                              levels = c("Low noise", "Moderate noise", "High noise"))) %>%
  group_by(ls_type, noise_level, sa_scenario, window_x) %>%
  summarise(mean_r2 = round(mean(mod_r2), 2))

ggplot(r2_df, aes(x = window_x, y = sa_scenario, fill = mean_r2)) +
  geom_tile() +
  scale_fill_viridis_c(expression(R^2)) +
  labs(x = "Window % of landscape", y = "") +
  facet_grid(noise_level ~ ls_type)
```



Session Info

```
session <- devtools::session_info()
session[[1]]
```

```
##  setting  value
##  version  R version 3.5.1 (2018-07-02)
##  os       Windows 10 x64
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United Kingdom.1252
##  ctype    English_United Kingdom.1252
##  tz       Europe/London
##  date     2018-11-08
```

```
session[[2]] %>% kable
```

	package	ondiskversion	loadedversion	path	loa
assertthat	assertthat	0.2.0	0.2.0	C:/Users/lg1u16/R/win-library/3.5/assertthat	C:/
backports	backports	1.1.2	1.1.2	C:/Users/lg1u16/R/win-library/3.5/backports	C:/
base64enc	base64enc	0.1.3	0.1-3	C:/Users/lg1u16/R/win-library/3.5/base64enc	C:/
bindr	bindr	0.1.1	0.1.1	C:/Users/lg1u16/R/win-library/3.5/bindr	C:/
bindrcpp	bindrcpp	0.2.2	0.2.2	C:/Users/lg1u16/R/win-library/3.5/bindrcpp	C:/
broom	broom	0.5.0	0.5.0	C:/Users/lg1u16/R/win-library/3.5/broom	C:/
callr	callr	3.0.0	3.0.0	C:/Users/lg1u16/R/win-library/3.5/callr	C:/
captioner	captioner	2.2.3.9000	2.2.3.9000	C:/Users/lg1u16/R/win-library/3.5/captioner	C:/
cellranger	cellranger	1.1.0	1.1.0	C:/Users/lg1u16/R/win-library/3.5/cellranger	C:/
checkmate	checkmate	1.8.5	1.8.5	C:/Users/lg1u16/R/win-library/3.5/checkmate	C:/
cli	cli	1.0.1	1.0.1	C:/Users/lg1u16/R/win-library/3.5/cli	C:/
codetools	codetools	0.2.15	0.2-15	C:/Program Files/R/R-3.5.1/library/codetools	C:/
colorspace	colorspace	1.3.2	1.3-2	C:/Users/lg1u16/R/win-library/3.5/colorspace	C:/
cowplot	cowplot	0.9.3	0.9.3	C:/Users/lg1u16/R/win-library/3.5/cowplot	C:/
crayon	crayon	1.3.4	1.3.4	C:/Users/lg1u16/R/win-library/3.5/crayon	C:/
debugme	debugme	1.1.0	1.1.0	C:/Users/lg1u16/R/win-library/3.5/debugme	C:/
desc	desc	1.2.0	1.2.0	C:/Users/lg1u16/R/win-library/3.5/desc	C:/
devtools	devtools	2.0.0	2.0.0	C:/Users/lg1u16/R/win-library/3.5/devtools	C:/
digest	digest	0.6.18	0.6.18	C:/Users/lg1u16/R/win-library/3.5/digest	C:/
dplyr	dplyr	0.7.7	0.7.7	C:/Users/lg1u16/R/win-library/3.5/dplyr	C:/
evaluate	evaluate	0.12	0.12	C:/Users/lg1u16/R/win-library/3.5/evaluate	C:/
extrafont	extrafont	0.17	0.17	C:/Users/lg1u16/R/win-library/3.5/extrafont	C:/
extrafontdb	extrafontdb	1.0	1.0	C:/Users/lg1u16/R/win-library/3.5/extrafontdb	C:/
forcats	forcats	0.3.0	0.3.0	C:/Users/lg1u16/R/win-library/3.5/forcats	C:/
fs	fs	1.2.6	1.2.6	C:/Users/lg1u16/R/win-library/3.5/fs	C:/
ggplot2	ggplot2	3.1.0	3.1.0	C:/Users/lg1u16/R/win-library/3.5/ggplot2	C:/
glue	glue	1.3.0	1.3.0	C:/Users/lg1u16/R/win-library/3.5/glue	C:/
grainchanger	grainchanger	0.0.0.9000	0.0.0.9000	C:/Users/lg1u16/R/win-library/3.5/grainchanger	C:/
gtable	gtable	0.2.0	0.2.0	C:/Users/lg1u16/R/win-library/3.5/gtable	C:/
haven	haven	1.1.2	1.1.2	C:/Users/lg1u16/R/win-library/3.5/haven	C:/
hms	hms	0.4.2	0.4.2	C:/Users/lg1u16/R/win-library/3.5/hms	C:/
htmltools	htmltools	0.3.6	0.3.6	C:/Users/lg1u16/R/win-library/3.5/htmltools	C:/
httr	httr	1.3.1	1.3.1	C:/Users/lg1u16/R/win-library/3.5/httr	C:/
jsonlite	jsonlite	1.5	1.5	C:/Users/lg1u16/R/win-library/3.5/jsonlite	C:/
knitr	knitr	1.20	1.20	C:/Users/lg1u16/R/win-library/3.5/knitr	C:/

	package	ondiskversion	loadedversion	path	loa
labeling	labeling	0.3	0.3	C:/Users/lg1u16/R/win-library/3.5/labeling	C:/
landscapetools	landscapetools	0.4.0	0.4.0	C:/Users/lg1u16/R/win-library/3.5/landscapetools	C:/
lattice	lattice	0.20.35	0.20-35	C:/Program Files/R/R-3.5.1/library/lattice	C:/
lazyeval	lazyeval	0.2.1	0.2.1	C:/Users/lg1u16/R/win-library/3.5/lazyeval	C:/
lubridate	lubridate	1.7.4	1.7.4	C:/Users/lg1u16/R/win-library/3.5/lubridate	C:/
magrittr	magrittr	1.5	1.5	C:/Users/lg1u16/R/win-library/3.5/magrittr	C:/
memoise	memoise	1.1.0	1.1.0	C:/Users/lg1u16/R/win-library/3.5/memoise	C:/
modelr	modelr	0.1.2	0.1.2	C:/Users/lg1u16/R/win-library/3.5/modelr	C:/
munsell	munsell	0.5.0	0.5.0	C:/Users/lg1u16/R/win-library/3.5/munsell	C:/
nlme	nlme	3.1.137	3.1-137	C:/Program Files/R/R-3.5.1/library/nlme	C:/
NLMR	NLMR	0.3.2	0.3.2	C:/Users/lg1u16/R/win-library/3.5/NLMR	C:/
patchwork	patchwork	0.0.1	0.0.1	C:/Users/lg1u16/R/win-library/3.5/patchwork	C:/
pillar	pillar	1.3.0	1.3.0	C:/Users/lg1u16/R/win-library/3.5/pillar	C:/
pkgbuild	pkgbuild	1.0.2	1.0.2	C:/Users/lg1u16/R/win-library/3.5/pkgbuild	C:/
pkgconfig	pkgconfig	2.0.2	2.0.2	C:/Users/lg1u16/R/win-library/3.5/pkgconfig	C:/
pkgload	pkgload	1.0.1	1.0.1	C:/Users/lg1u16/R/win-library/3.5/pkgload	C:/
plyr	plyr	1.8.4	1.8.4	C:/Users/lg1u16/R/win-library/3.5/plyr	C:/
prettyunits	prettyunits	1.0.2	1.0.2	C:/Users/lg1u16/R/win-library/3.5/prettyunits	C:/
processx	processx	3.2.0	3.2.0	C:/Users/lg1u16/R/win-library/3.5/processx	C:/
ps	ps	1.2.0	1.2.0	C:/Users/lg1u16/R/win-library/3.5/ps	C:/
purrr	purrr	0.2.5	0.2.5	C:/Users/lg1u16/R/win-library/3.5/purrr	C:/
R6	R6	2.3.0	2.3.0	C:/Users/lg1u16/R/win-library/3.5/R6	C:/
raster	raster	2.8.6	2.8-6	C:/Users/lg1u16/R/win-library/3.5/raster	C:/
Rcpp	Rcpp	0.12.19	0.12.19	C:/Users/lg1u16/R/win-library/3.5/Rcpp	C:/
readr	readr	1.1.1	1.1.1	C:/Users/lg1u16/R/win-library/3.5/readr	C:/
readxl	readxl	1.1.0	1.1.0	C:/Users/lg1u16/R/win-library/3.5/readxl	C:/
remotes	remotes	2.0.1	2.0.1	C:/Users/lg1u16/R/win-library/3.5/remotes	C:/
reshape2	reshape2	1.4.3	1.4.3	C:/Users/lg1u16/R/win-library/3.5/reshape2	C:/
rlang	rlang	0.3.0.1	0.3.0.1	C:/Users/lg1u16/R/win-library/3.5/rlang	C:/
rmarkdown	rmarkdown	1.10	1.10	C:/Users/lg1u16/R/win-library/3.5/rmarkdown	C:/
rprojroot	rprojroot	1.3.2	1.3-2	C:/Users/lg1u16/R/win-library/3.5/rprojroot	C:/
rstudioapi	rstudioapi	0.8	0.8	C:/Users/lg1u16/R/win-library/3.5/rstudioapi	C:/
Rttf2pt1	Rttf2pt1	1.3.7	1.3.7	C:/Users/lg1u16/R/win-library/3.5/Rttf2pt1	C:/
rvest	rvest	0.3.2	0.3.2	C:/Users/lg1u16/R/win-library/3.5/rvest	C:/
scales	scales	1.0.0	1.0.0	C:/Users/lg1u16/R/win-library/3.5/scales	C:/
sessioninfo	sessioninfo	1.1.0	1.1.0	C:/Users/lg1u16/R/win-library/3.5/sessioninfo	C:/
sp	sp	1.3.1	1.3-1	C:/Users/lg1u16/R/win-library/3.5/sp	C:/
stringi	stringi	1.2.4	1.2.4	C:/Users/lg1u16/R/win-library/3.5/stringi	C:/
stringr	stringr	1.3.1	1.3.1	C:/Users/lg1u16/R/win-library/3.5/stringr	C:/
testthat	testthat	2.0.1	2.0.1	C:/Users/lg1u16/R/win-library/3.5/testthat	C:/
tibble	tibble	1.4.2	1.4.2	C:/Users/lg1u16/R/win-library/3.5/tibble	C:/
tidyr	tidyr	0.8.2	0.8.2	C:/Users/lg1u16/R/win-library/3.5/tidyr	C:/
tidyselect	tidyselect	0.2.5	0.2.5	C:/Users/lg1u16/R/win-library/3.5/tidyselect	C:/
tidyverse	tidyverse	1.2.1	1.2.1	C:/Users/lg1u16/R/win-library/3.5/tidyverse	C:/
usethis	usethis	1.4.0	1.4.0	C:/Users/lg1u16/R/win-library/3.5/usethis	C:/
viridisLite	viridisLite	0.3.0	0.3.0	C:/Users/lg1u16/R/win-library/3.5/viridisLite	C:/
withr	withr	2.1.2	2.1.2	C:/Users/lg1u16/R/win-library/3.5/withr	C:/
xml2	xml2	1.2.0	1.2.0	C:/Users/lg1u16/R/win-library/3.5/xml2	C:/
yaml	yaml	2.2.0	2.2.0	C:/Users/lg1u16/R/win-library/3.5/yaml	C:/