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(plyr)
library(landscapetools)
library(NLMR)
library(R.utils)
library(tidyverse)
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.")
```

Methods

Moving window approach

```
ls <- nlm_random(10, 10) %>% as.data.frame(xy = TRUE)
w1 <- data.frame(x = rep(c(0.5, 1.5, 2.5), times = 3),
                 y = rep(c(7.5, 8.5, 9.5), each = 3))
w2 <- mutate(w1, x = x+1)
w3 <- mutate(w2, x = x+1)

w1_plot <- ggplot() +
  geom_raster(data = ls, aes(x = x, y = y, fill = layer)) +
  geom_raster(data = w1, aes(x = x, y = y), fill = "grey", alpha = 0.7) +
  annotate("text", x = 1.5, y = 8.5,
           label = "italic(f(x))", parse = TRUE, size = 2.5) +
  annotate("text", x = 0, y = 11.2, hjust = 0,
           label = "Predictor grain", size = 2.5) +
  annotate("text", x = 0, y = 12.6, hjust = 0,
           label = "Scale of effect", size = 2.5) +
```

```
  annotate("text", x = 0, y = 14, hjust = 0,
           label = "Response grain", size = 2.5) +
  geom_segment(aes(x=0, xend=1, y=10.5, yend=10.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both")) +
  geom_segment(aes(x=0, xend=3, y=11.9, yend=11.9), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both")) +
  geom_segment(aes(x=0, xend=10, y=13.3, yend=13.3), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both")) +
  scale_fill_viridis_c() +
  coord_equal() +
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.border = element_blank(),
        legend.position = "none")

w2_plot <- ggplot() +
  geom_raster(data = ls, aes(x = x, y = y, fill = layer)) +
  geom_raster(data = w2, aes(x = x, y = y), fill = "grey", alpha = 0.7) +
  annotate("text", x = 2.5, y = 8.5,
           label = "italic(f(x))", parse = TRUE, size = 2.5) +
  annotate("text", x = 0, y = 11.2, hjust = 0,
           label = "Predictor grain", size = 2.5, colour = "white") +
  annotate("text", x = 0, y = 12.6, hjust = 0,
           label = "Scale of effect", size = 2.5, colour = "white") +
  annotate("text", x = 0, y = 14, hjust = 0,
           label = "Response grain", size = 2.5, colour = "white") +
  geom_segment(aes(x=0, xend=1, y=10.5, yend=10.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both"),
               colour = "white") +
  geom_segment(aes(x=0, xend=3, y=11.9, yend=11.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both"),
               colour = "white") +
  geom_segment(aes(x=0, xend=10, y=13.3, yend=12.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both"),
               colour = "white") +
  scale_fill_viridis_c() +
  coord_equal() +
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.border = element_blank(),
        legend.position = "none")

w3_plot <- ggplot() +
  geom_raster(data = ls, aes(x = x, y = y, fill = layer)) +
  geom_raster(data = w3, aes(x = x, y = y), fill = "grey", alpha = 0.7) +
  annotate("text", x = 3.5, y = 8.5,
           label = "italic(f(x))",
           parse = TRUE,
           size = 2.5) +
```

```
  annotate("text", x = 0, y = 11.2, hjust = 0,
           label = "Predictor grain", size = 2.5, colour = "white") +
  annotate("text", x = 0, y = 12.6, hjust = 0,
           label = "Scale of effect", size = 2.5, colour = "white") +
  annotate("text", x = 0, y = 14, hjust = 0,
           label = "Response grain", size = 2.5, colour = "white") +
  geom_segment(aes(x=0, xend=1, y=10.5, yend=10.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both"),
               colour = "white") +
  geom_segment(aes(x=0, xend=3, y=11.9, yend=11.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both"),
               colour = "white") +
  geom_segment(aes(x=0, xend=10, y=13.3, yend=12.5), size = 0.5,
               arrow = arrow(length = unit(0.1, "cm"), ends = "both"),
               colour = "white") +
  scale_fill_viridis_c() +
  coord_equal() +
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.border = element_blank(),
        legend.position = "none")

plot_grid(w1_plot, w2_plot, w3_plot, nrow = 1, axis = "b")
```

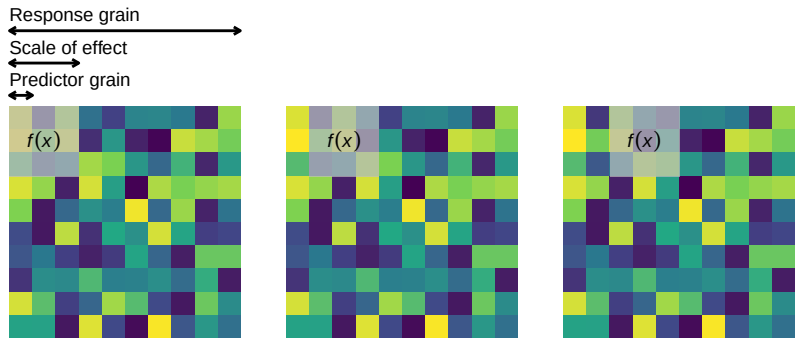

Figure AI. 1: Graphical representation of the moving window method. When the moving window is defined, three aspects of scale need to be considered. Predictor grain is the characteristic spatial scale of the predictor variable, i.e. the resolution of the environmental data; scale of effect determines 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).

Simulation One: continuous variables

This simulation shows the use of the moving window based spatial structure measure for continuous variables which have finer scale variation than the resolution of analysis. This could be, for example, topographic or microclimate variation.

Simulated landscapes

We simulated landscapes using the fractal Brownian motion method (`nlm_fbm`) from the `NLMR` package [@Sciaini2018]. We simulated landscapes with three levels of spatial autocorrelation, and 100 replicates for each of these. The spatial autocorrelation of the landscape is controlled by the `fract_dim` parameter where a value close to zero is a rough landscape (random), and a value of one is a smooth landscape (spatially autocorrelated). We generated landscapes of 65 x 65 cells using fractal dimension = 0.1, 0.5, 1.

```
nrows <- ncols <- 65
fract_dim <- c(0.1, 0.5, 1)
radius <- c(1, 10, 20)
reps <- 1:100

# set up a table with all combinations of parameters
params <- expand.grid(ncol = ncols, nrow = nrows, fract_dim = fract_dim, reps = reps)

# simulate 65 x 65 landscapes with variable roughness
sim_ls <- apply(params, 1, function(x) {
  ls <- nlm_fbm(ncol = x['ncol'],
          nrow = x['nrow'],
          fract_dim = as.numeric(x['fract_dim']),
          rescale = TRUE)
  ls_df <- raster::as.data.frame(ls, xy = TRUE) %>%
    cbind(x %>% as.data.frame %>% t, row.names = NULL)
  return(list(params = x, ls = ls, ls_df = ls_df))
})
```

Calculate MW environmental heterogeneity

We used the `winmove` function from the `grainchanger` package to gain a moving-window based range and variance of the continuous variable at 3 different window radii: 1 = small; 7 = medium; 17 = large. The window sizes represent the appropriate scale of effect of the continuous variable on the response. For the MW variance calculation, we scaled the landscapes to have a mean of 0 and variance of 1. For the MW range calculation we min-max scaled the landscapes to take values between 0 and 1. In both cases, the metric when calculated at the landscape scale without the moving window is 1. The `create_torus()` function (also in the `grainchanger` package) allows us to buffer a given landscape by a given radius, creating the effect of a torus, and thus avoiding issues of edge effects.

```
cont_res <- ldply(sim_ls, function(x) {
  ls <- x$ls
  ldply(radius, function(r) {
    ls_pad <- create_torus(ls, r)
    # scale to make variance identical across landscapes
    mw_var <- winmove(raster::scale(ls_pad), radius = r, type = "rectangle", fn = "var")
    mw_var <- raster::trim(mw_var)
    mw_range <- winmove(ls_pad, radius = r, type = "rectangle", fn = "var_range")
    mw_range <- raster::trim(mw_range)
```

```
    out <- data.frame(t(x$params),
                      radius = r,
                      val = c(mw_var %>% values %>% mean,
                              mw_range %>% values %>% mean),
                      measure = c("MW variance", "MW range"))
  })
}) %>%
  mutate(radius = factor(radius,
                         labels = c("Small: 1",
                                    "Medium: 10",
                                    "Large: 20")),
         fract_dim = factor(paste0("Fractal dimension = ", fract_dim)))
```

Simulation Two: categorical variables

This simulation will show the use of the moving window-based spatial structure measure for categorical variables which have finer scale variation than the resolution of analysis. This is appropriate for more classical landscape ecology questions about habitat structure. On these landscapes we calculated Shannon evenness (MW Shannon) and standardised patch richness (MW richness).

Simulated landscapes

We will use the simulated landscapes from the previous example, and use the `util_classify` function to classify into equal proportions of five habitat types. When calculating the landscape scale metrics (MW Shannon and MW richness) without a moving window, the value in both cases is 1.

Calculate MW environmental heterogeneity

We used the `winmove` function from the `grainchanger` package to gain moving-window based environmental heterogeneity measures (MW Shannon and MW richness) of the categorical variable at 3 different window radius sizes: 1 = small; 7 = medium; 17 = large. The window sizes represent the appropriate scale of effect of the categorical variable on the response. The `create_torus()` function (also in the `grainchanger` package) allows us to buffer a given landscape by a given radius, creating the effect of a torus, and thus avoiding issues of edge effects.

```
cat_res <- ldply(sim_ls, function(x) {
  ls <- util_classify(x$ls, weighting = rep(1/5, 5))
  ldply(radius, function(r) {
    ls_pad <- create_torus(ls, r)
    # calculate shannon evenness
    mw_shei <- winmove(ls_pad,
                       radius = r,
                       type = "rectangle",
                       fn = "diversity",
                       lc_class = 0:4)
    mw_shei <- raster::trim(mw_shei)
    # calculate patch richness index (pr / # landcovers)
    mw_pr <- winmove(ls_pad,
                     radius = r,
                     type = "rectangle",
                     fn = "diversity",
```

```
                    index = "pr",
                    lc_class = 0:4)
    mw_pr <- raster::trim(mw_pr/5) # divide by number of landcovers to get value from 0:1
    out <- data.frame(t(x$params),
                      radius = r,
                      val = c(mw_shei %>% values %>% mean,
                              mw_pr %>% values %>% mean),
                      measure = c("MW Shannon", "MW richness"))
  })
}) %>%
  mutate(radius = factor(radius,
                         labels = c("Small: 1", "Medium: 10", "Large: 20")),
         fract_dim = factor(paste0("Fractal dimension = ", fract_dim)))
```

Results

Plots

```
# combine all runs into one dataframe and only take the first replicate of each
# combination for plotting
cont_ls_df <- ldply(sim_ls, function(x) x$ls_df) %>%
  filter(reps == 1) %>%
  mutate(ls_type = "Continuous")


cat_ls_df <- ldply(sim_ls, function(x) {
  ls <- util_classify(x$ls, weighting = rep(1/5, 5))
  ls_df <- raster::as.data.frame(ls, xy = TRUE)
  return(data.frame(t(x$params), ls_df))
}) %>%
  filter(reps == 1) %>%
  mutate(ls_type = "Categorical")


cont_ls_plot <- ggplot(cont_ls_df, aes(x = x, y = y, fill = layer)) +
  geom_raster() +
  coord_equal() +
  scale_fill_viridis_c() +
  theme(axis.text = element_blank(), axis.title = element_blank(),
        axis.line = element_blank(), axis.ticks = element_blank()) +
  facet_grid(fract_dim~ls_type) +
  theme(axis.text = element_blank(),
        axis.title = element_blank(),
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        strip.background = element_blank(),
        strip.text.y = element_blank(),
        legend.position = "none",
        panel.border = element_blank())

cat_ls_plot <- ggplot(cat_ls_df, aes(x = x, y = y, fill = layer)) +
```

```
  geom_raster() +
  coord_equal() +
  scale_fill_viridis_c() +
  facet_grid(fract_dim~ls_type) +
  theme(axis.text = element_blank(),
        axis.title = element_blank(),
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        strip.background = element_blank(),
        strip.text.y = element_blank(),
        legend.position = "none",
        panel.border = element_blank())

res_df <- bind_rows(cont_res, cat_res) %>%
  mutate(measure = factor(measure,
                          levels = c("MW range",
                                     "MW variance",
                                     "MW richness",
                                     "MW Shannon")))

res_plot <- ggplot(res_df, aes(x = radius, y = val, fill = measure)) +
  geom_boxplot(lwd = 0.1, outlier.size = 0.5) + coord_flip() +
  scale_fill_grey("") +
  facet_wrap(~fract_dim, ncol = 1) +
  xlab("Moving window radius") + ylab("")

plot_grid(cont_ls_plot, cat_ls_plot, res_plot,
          align = 'h',
          rel_widths = c(1,1,3),
          nrow = 1,
          labels = c("a)", "", "b)"),
          label_size = 7)
```

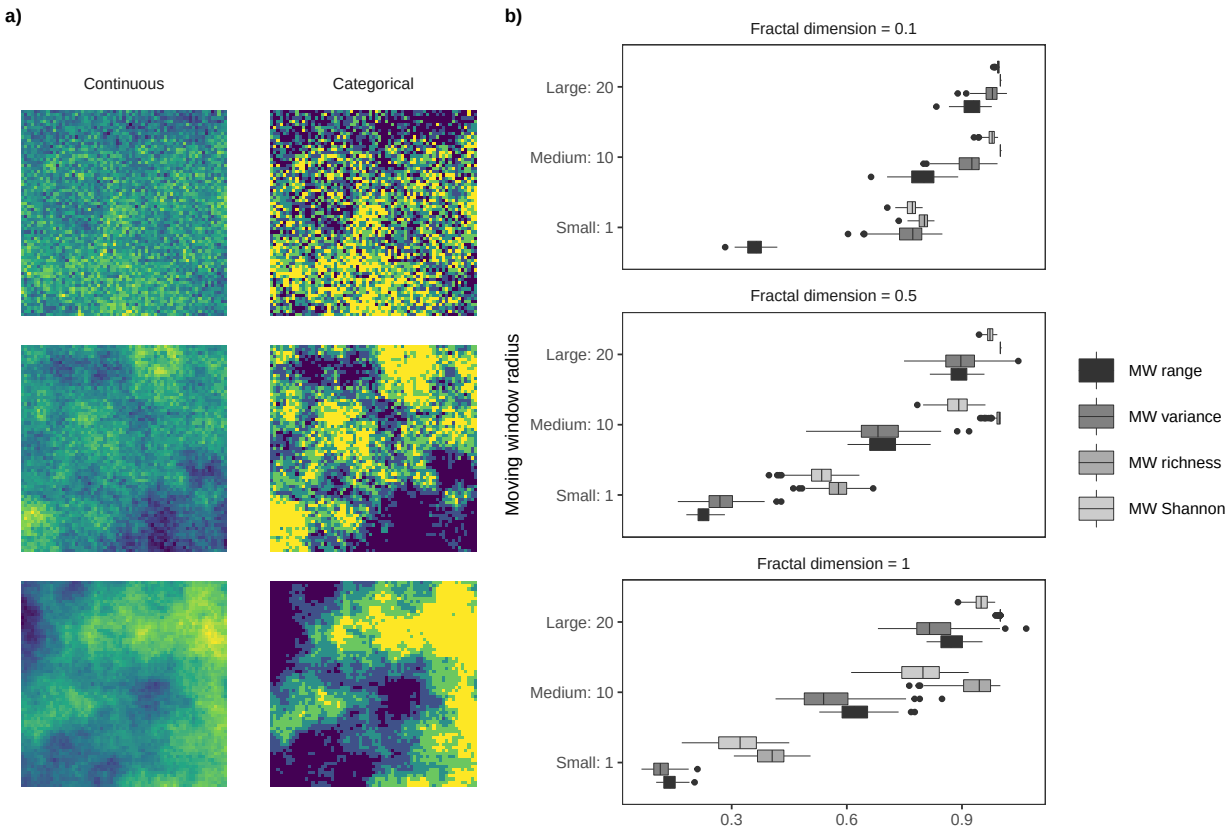

Figure AI. 2: Example landscapes of a continuous and categorical variable for each level of spatial autocorrelation (a). Results of the moving window calculations for simulated landscapes of three levels of spatial autocorrelation (b). MW variance was calculated on a continuous environmental variable with mean = 0 and variance = 1. MW range was calculated on a continuous landscape with values in [0, 1]. MW Shannon and MW richness were calculated on a categorical landscape with five categories of equal proportions. All measures were calculated within moving windows were at three scales-of-effect. Note that when calculated without a moving window, the landscape scale value is 1 in all cases.

Analysis of variance

We used a two-way analysis of variance in order to understand the effect of spatial structure in the landscape and the size of the moving window on the output measure. We focus on the effect size rather than the statistical significance due to the inflated sample size in simulated data.

```
res_df %>%
  group_by(measure) %>%
  nest() %>%
  mutate(mod = map(data, function(x) aov(val ~ fract_dim * radius, data = x)),
         res = map(mod, tidy)) %>%
  select(measure, res) %>%
  unnest() %>%
  select(-p.value) %>%
  kable(digits = 3)
```

measure	term	df	sumsq	meansq	statistic

measure	term	df	sumsq	meansq	statistic
MW variance	fract_dim	2	23.201	11.600	3330.938
MW variance	radius	2	41.464	20.732	5952.934
MW variance	fract_dim:radius	4	7.180	1.795	515.405
MW variance	Residuals	891	3.103	0.003	NA
MW range	fract_dim	2	3.385	1.692	1393.561
MW range	radius	2	67.898	33.949	27956.466
MW range	fract_dim:radius	4	0.772	0.193	159.009
MW range	Residuals	891	1.082	0.001	NA
MW Shannon	fract_dim	2	7.668	3.834	2504.587
MW Shannon	radius	2	31.590	15.795	10318.398
MW Shannon	fract_dim:radius	4	4.237	1.059	691.969
MW Shannon	Residuals	891	1.364	0.002	NA
MW richness	fract_dim	2	3.519	1.759	2148.834
MW richness	radius	2	31.384	15.692	19165.682
MW richness	fract_dim:radius	4	4.519	1.130	1379.809
MW richness	Residuals	891	0.730	0.001	NA

Table AI. 1: Two-way analysis of variance for each MW measure. Note p-values are not displayed because they are artificially inflated and thus meaningless in simulation studies. We therefore concentrate on effect size.

Session Info

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

```
##  setting  value
##  version  R version 3.5.0 (2018-04-23)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United Kingdom.1252
##  tz       Europe/London
##  date     2018-06-21
```

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

package	*	version	date	source
assertthat		0.2.0	2017-04-11	CRAN (R 3.5.0)
backports		1.1.2	2017-12-13	CRAN (R 3.5.0)
base	*	3.5.0	2018-04-23	local
bindr		0.1.1	2018-03-13	CRAN (R 3.5.0)
bindrcpp	*	0.2.2	2018-03-29	CRAN (R 3.5.0)
broom	*	0.4.4	2018-03-29	CRAN (R 3.5.0)
captioner	*	2.2.3.9000	2018-06-19	Github (adletaw/captioner@5f2b435)
cellranger		1.1.0	2016-07-27	CRAN (R 3.5.0)
checkmate		1.8.5	2017-10-24	CRAN (R 3.5.0)

package	*	version	date	source
cli		1.0.0	2017-11-05	CRAN (R 3.5.0)
codetools		0.2-15	2016-10-05	CRAN (R 3.5.0)
colorspace		1.3-2	2016-12-14	CRAN (R 3.5.0)
compiler		3.5.0	2018-04-23	local
cowplot	*	0.9.2	2017-12-17	CRAN (R 3.5.0)
crayon		1.3.4	2017-09-16	CRAN (R 3.5.0)
datasets	*	3.5.0	2018-04-23	local
devtools		1.13.5	2018-02-18	CRAN (R 3.5.0)
digest		0.6.15	2018-01-28	CRAN (R 3.5.0)
dplyr	*	0.7.4	2017-09-28	CRAN (R 3.5.0)
evaluate		0.10.1	2017-06-24	CRAN (R 3.5.0)
extrafont		0.17	2014-12-08	CRAN (R 3.5.0)
extrafontdb		1.0	2012-06-11	CRAN (R 3.5.0)
forcats	*	0.3.0	2018-02-19	CRAN (R 3.5.0)
foreign		0.8-70	2017-11-28	CRAN (R 3.5.0)
ggplot2	*	2.2.1.9000	2018-05-23	Github (tidyverse/ggplot2@eecc450)
glue		1.2.0	2017-10-29	CRAN (R 3.5.0)
grainchanger	*	0.0.0.9000	2018-06-06	local (laurajanegraham/grainchanger@NA)
graphics	*	3.5.0	2018-04-23	local
grDevices	*	3.5.0	2018-04-23	local
grid		3.5.0	2018-04-23	local
gtable		0.2.0	2016-02-26	CRAN (R 3.5.0)
haven		1.1.1	2018-01-18	CRAN (R 3.5.0)
highr		0.6	2016-05-09	CRAN (R 3.5.0)
hms		0.4.2	2018-03-10	CRAN (R 3.5.0)
htmltools		0.3.6	2017-04-28	CRAN (R 3.5.0)
httr		1.3.1	2017-08-20	CRAN (R 3.5.0)
jsonlite		1.5	2017-06-01	CRAN (R 3.5.0)
knitr	*	1.20	2018-02-20	CRAN (R 3.5.0)
labeling		0.3	2014-08-23	CRAN (R 3.5.0)
landscapetools	*	0.3.0	2018-03-28	CRAN (R 3.5.0)
lattice		0.20-35	2017-03-25	CRAN (R 3.5.0)
lazyeval		0.2.1	2017-10-29	CRAN (R 3.5.0)
lubridate		1.7.4	2018-04-11	CRAN (R 3.5.0)
magrittr		1.5	2014-11-22	CRAN (R 3.5.0)
memoise		1.1.0	2017-04-21	CRAN (R 3.5.0)
methods	*	3.5.0	2018-04-23	local
mnormt		1.5-5	2016-10-15	CRAN (R 3.5.0)
modelr		0.1.2	2018-05-11	CRAN (R 3.5.0)
munsell		0.4.3	2016-02-13	CRAN (R 3.5.0)
nlme		3.1-137	2018-04-07	CRAN (R 3.5.0)
NLMR	*	0.3.0	2018-04-03	CRAN (R 3.5.0)
parallel		3.5.0	2018-04-23	local
pillar		1.2.2	2018-04-26	CRAN (R 3.5.0)
pkgconfig		2.0.1	2017-03-21	CRAN (R 3.5.0)
plyr	*	1.8.4	2016-06-08	CRAN (R 3.5.0)
psych		1.8.4	2018-05-06	CRAN (R 3.5.0)
purrr	*	0.2.4	2017-10-18	CRAN (R 3.5.0)
R.methodsS3	*	1.7.1	2016-02-16	CRAN (R 3.5.0)
R.oo	*	1.22.0	2018-04-22	CRAN (R 3.5.0)
R.utils	*	2.6.0	2017-11-05	CRAN (R 3.5.0)
R6		2.2.2	2017-06-17	CRAN (R 3.5.0)

package	*	version	date	source
RandomFields		3.1.50	2017-04-17	CRAN (R 3.5.0)
RandomFieldsUtils		0.3.25	2017-04-14	CRAN (R 3.5.0)
raster	*	2.6-7	2017-11-13	CRAN (R 3.5.0)
Rcpp		0.12.16	2018-03-13	CRAN (R 3.5.0)
readr	*	1.1.1	2017-05-16	CRAN (R 3.5.0)
readxl		1.1.0	2018-04-20	CRAN (R 3.5.0)
reshape2		1.4.3	2017-12-11	CRAN (R 3.5.0)
rlang		0.2.0	2018-02-20	CRAN (R 3.5.0)
rmarkdown		1.9	2018-03-01	CRAN (R 3.5.0)
rprojroot		1.3-2	2018-01-03	CRAN (R 3.5.0)
rstudioapi		0.7	2017-09-07	CRAN (R 3.5.0)
Rttf2pt1		1.3.6	2018-02-22	CRAN (R 3.5.0)
rvest		0.3.2	2016-06-17	CRAN (R 3.5.0)
scales		0.5.0	2017-08-24	CRAN (R 3.5.0)
sp	*	1.2-7	2018-01-19	CRAN (R 3.5.0)
stats	*	3.5.0	2018-04-23	local
stringi		1.1.7	2018-03-12	CRAN (R 3.5.0)
stringr	*	1.3.1	2018-05-10	CRAN (R 3.5.0)
tibble	*	1.4.2	2018-01-22	CRAN (R 3.5.0)
tidyr	*	0.8.0	2018-01-29	CRAN (R 3.5.0)
tidyselect		0.2.4	2018-02-26	CRAN (R 3.5.0)
tidyverse	*	1.2.1	2017-11-14	CRAN (R 3.5.0)
tools		3.5.0	2018-04-23	local
utils	*	3.5.0	2018-04-23	local
viridisLite		0.3.0	2018-02-01	CRAN (R 3.5.0)
withr		2.1.2	2018-05-15	Github (jimhester/withr@79d7b0d)
xml2		1.2.0	2018-01-24	CRAN (R 3.5.0)
yaml		2.1.19	2018-05-01	CRAN (R 3.5.0)