Design Demo

L Marshall

08/05/2019

Introduction

This document gives a short demonstration of the usage and functionality of the design package to date. I include some details of future plans as well as points of query. The plot colours are user specified in these examples so I can test out colour schemes to use as the defaults.

Implementation Decisions

I decided to store the region object inside the design object as this does not in fact increase the memory usage much as it is essentially a pointer that is stored not a duplicate of the object. I therefore also plan to store a coverage grid object inside the design object too. This means that only one coverage grid can be associated with a design.

While I will set it up so that the coverage grid can be generated inside either the make.design function or the run(design) function I will advise against this as doing so will potentially result in multiple instances of identical coverage grid objects.

```r
#Demonstration of memory useage in R

#Please restart your R session
library(pryr)
```

```
## Registered S3 method overwritten by 'pryr':
##   method      from
##   print.bytes Rcpp
```

```r
mem_used()
```

```
## 33.4 MB
```

```r
x <- rep(123456789, 1000000)
mem_used()
```

```
## 41.4 MB
```

```r
y <- rep(123456789, 1000000)
mem_used()
```

```
## 49.4 MB
```

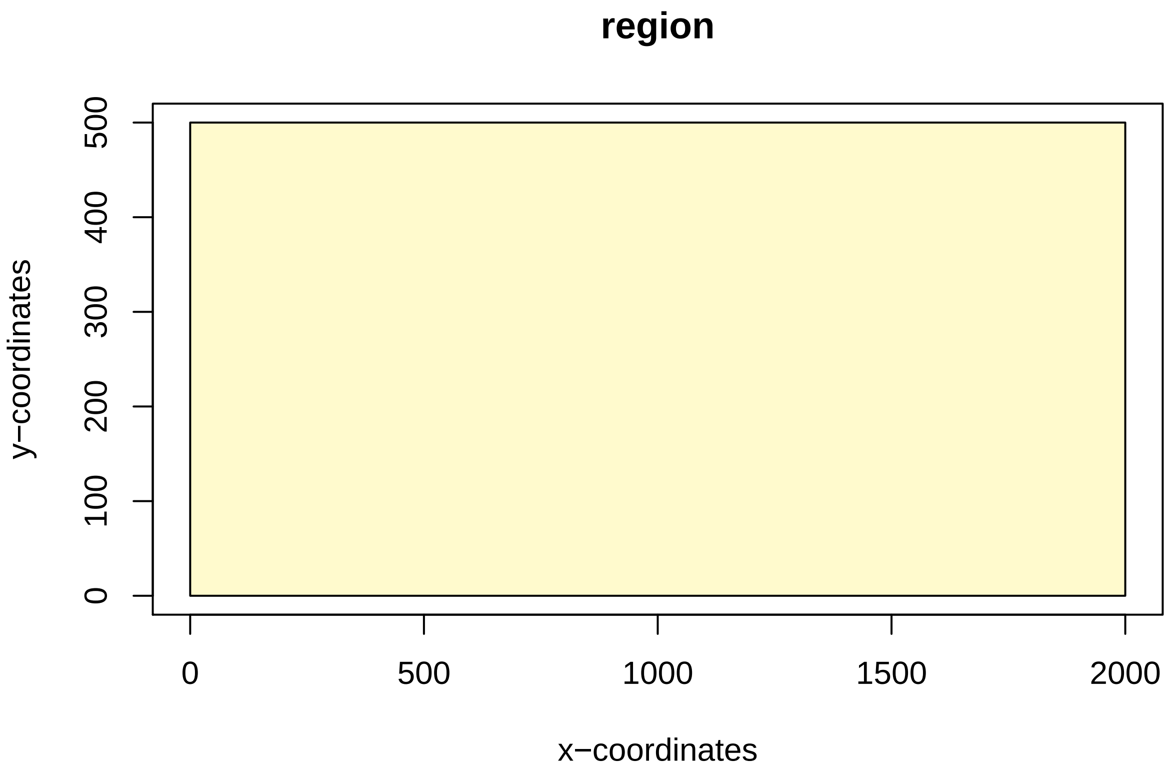
```r
z <- x
mem_used()
```

```
## 49.4 MB
```

Default Options

The design package will make a default region with the same dimensions as that in DSsim.

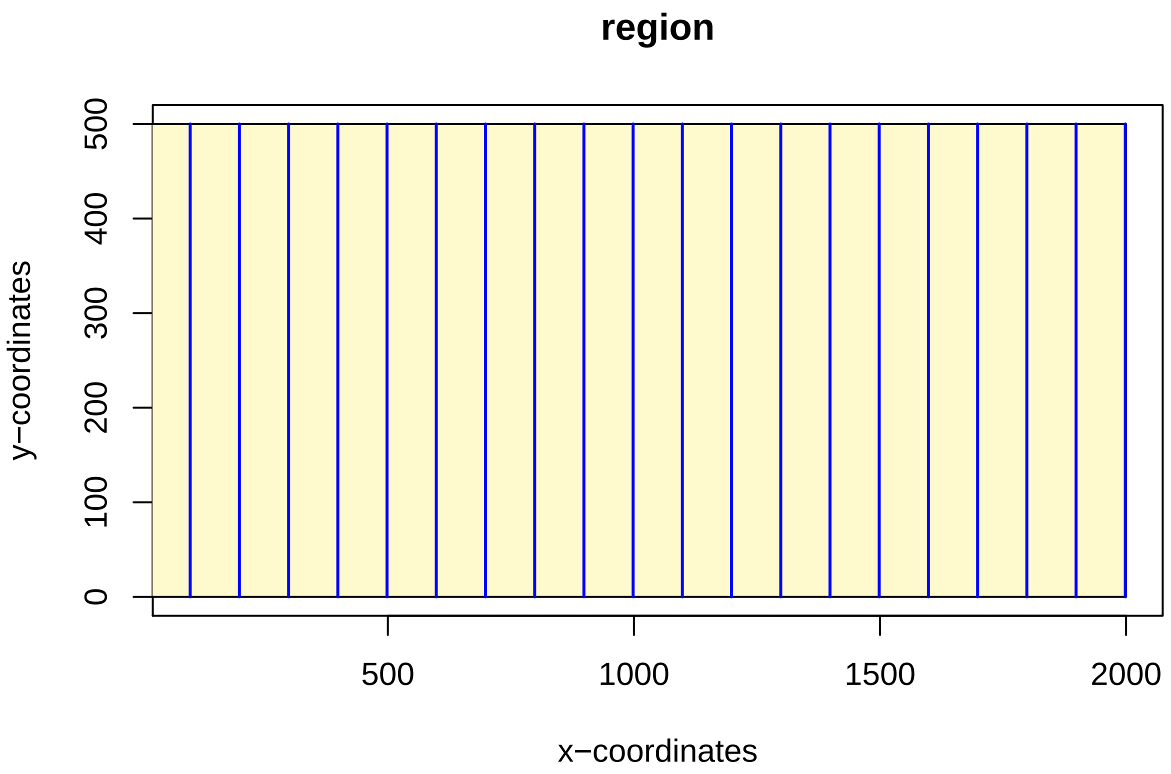
```r
library(dsuds)
region <- make.region()
plot(region, col = "lemonchiffon")
```



The default design is 20 systematic parallel lines with a design angle of 0 (design angle argument need correcting - the transects are what I wanted but I think this should be considered a design angle of 90 not 0?). See later on in the mixed design section for more discussion.

I have yet to implement the summary methods for the line transect objects so I have included details about what I will show.

```r
design <- make.design(region)
transects <- generate.transects(design)
plot(region, transects, region.col = "lemonchiffon", col = 4, lwd = 1.5)
```

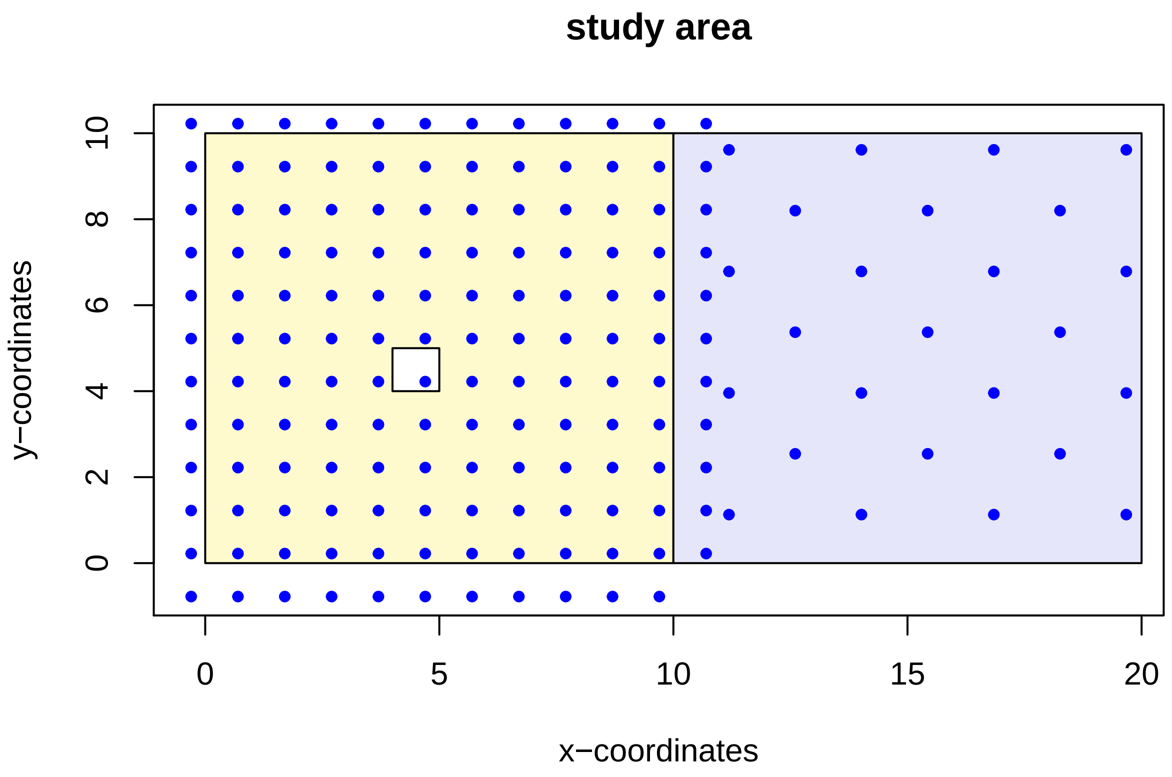


```r
#I have yet to implement summary functions
#The plan is to display the following for line transect surveys:
# - design
# - number of samplers per strata
# - total transect length per strata
# - coverage (proportion of each strata sampled by transects)
# - effort allocation values
# - spacing
# - design.angle
# - edge protocol
# - bounding shape
```

Point transects

I have implemented the systematic point transect design. This design can have different design arguments such as spacing, design.angle, no.samplers and plus/minus sampling for each strata. I have yet to implement the summary methods for the point transect objects so I have included details about what I will show.

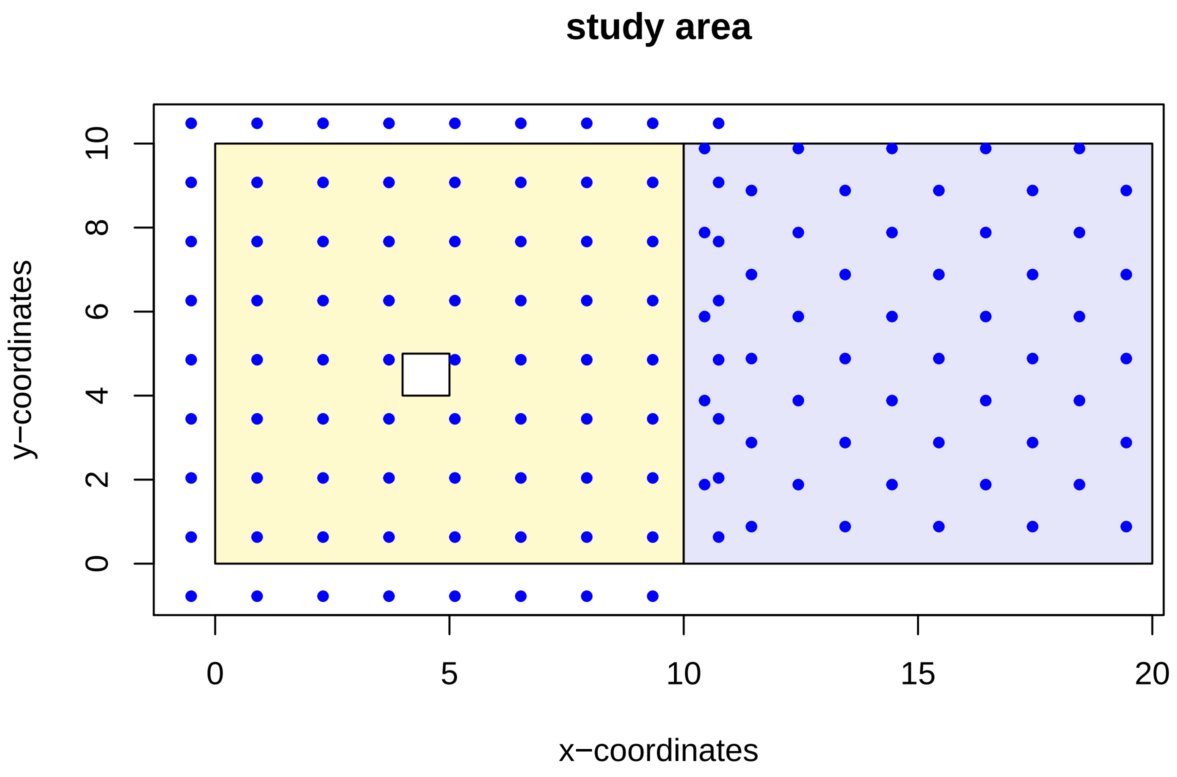
```
region <- make.region("study area", c("A", "B"), shape = "/Users/lhm/Documents/Work/design package/sf i
design <- make.design(region, transect.type = "point",
                      design = "systematic",
                      spacing = c(1,2),
                      edge.protocol = c("plus", "minus"),
                      design.angle = c(0,45))
transects <- generate.transects(design)
plot(region, transects, region.col = c("lemonchiffon", "lavender"), col = 4)
```



```
#I have yet to implement summary functions
#The plan is to display the following for point transect surveys:
# - design
# - number of samplers per strata
# - coverage (proportion of each strata sampled by transects)
# - effort allocation values
# - spacing
# - design.angle
# - edge protocol
```

The next example uses the desired number of samplers to determine the spacing. In this example the spacing is based on a calculation involving the strata area however in the case of plus sampling should this calculation be based on the buffered survey region? Otherwise it will generate many more than the desired number of points.

```r
design <- make.design(region, "point", "systematic",
                      no.samplers = c(50,50),
                      edge.protocol = c("plus", "minus"),
                      design.angle = c(0,45))
transects <- generate.transects(design)
plot(region, transects, region.col = c("lemonchiffon", "lavender"), col = 4)
```

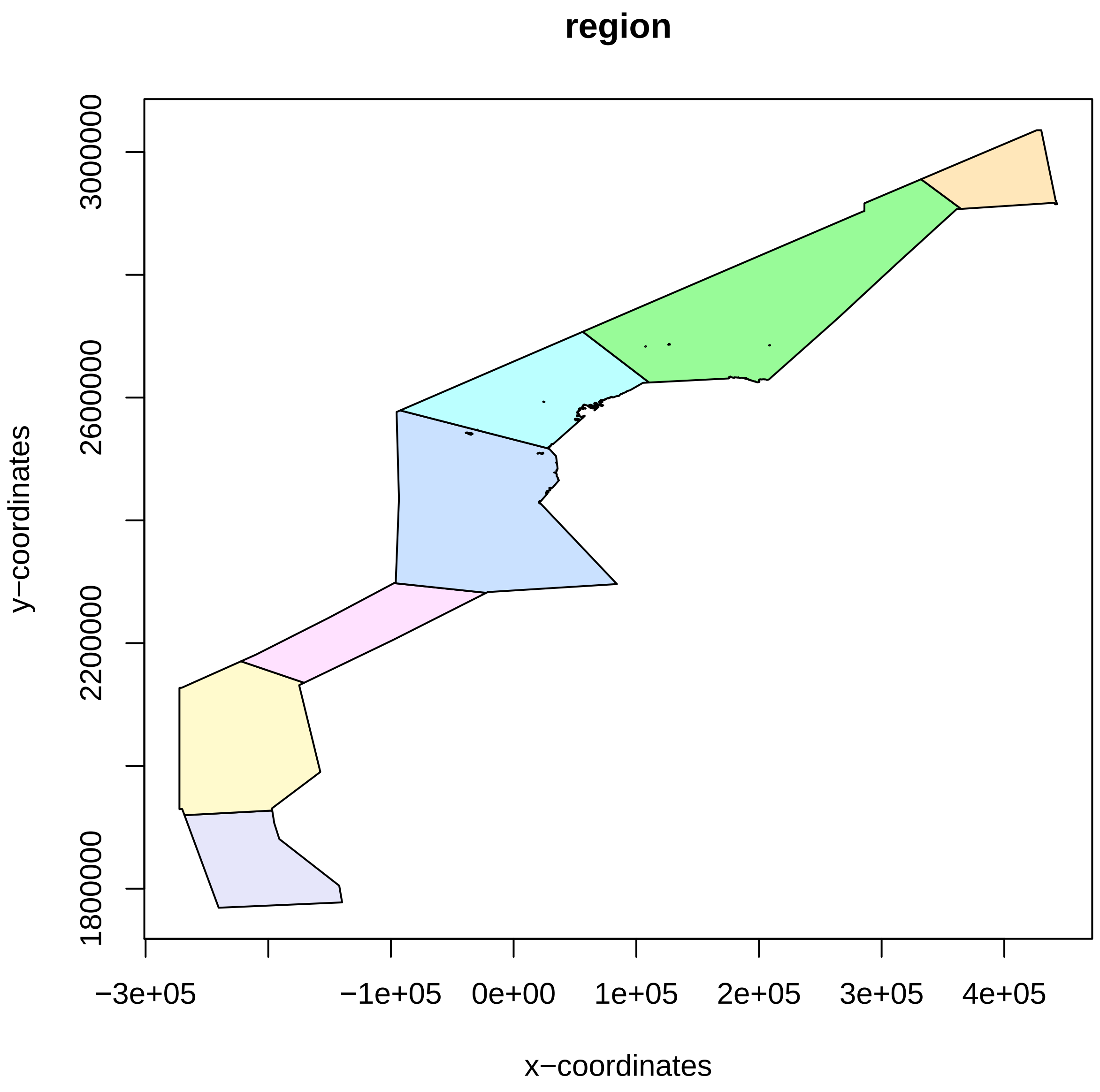


Example with Multiple Strata

This next example contains multiple strata. I have not yet coded any nice default colours so I'm setting these up manually. Please let me know what you think of the colours as defaults.

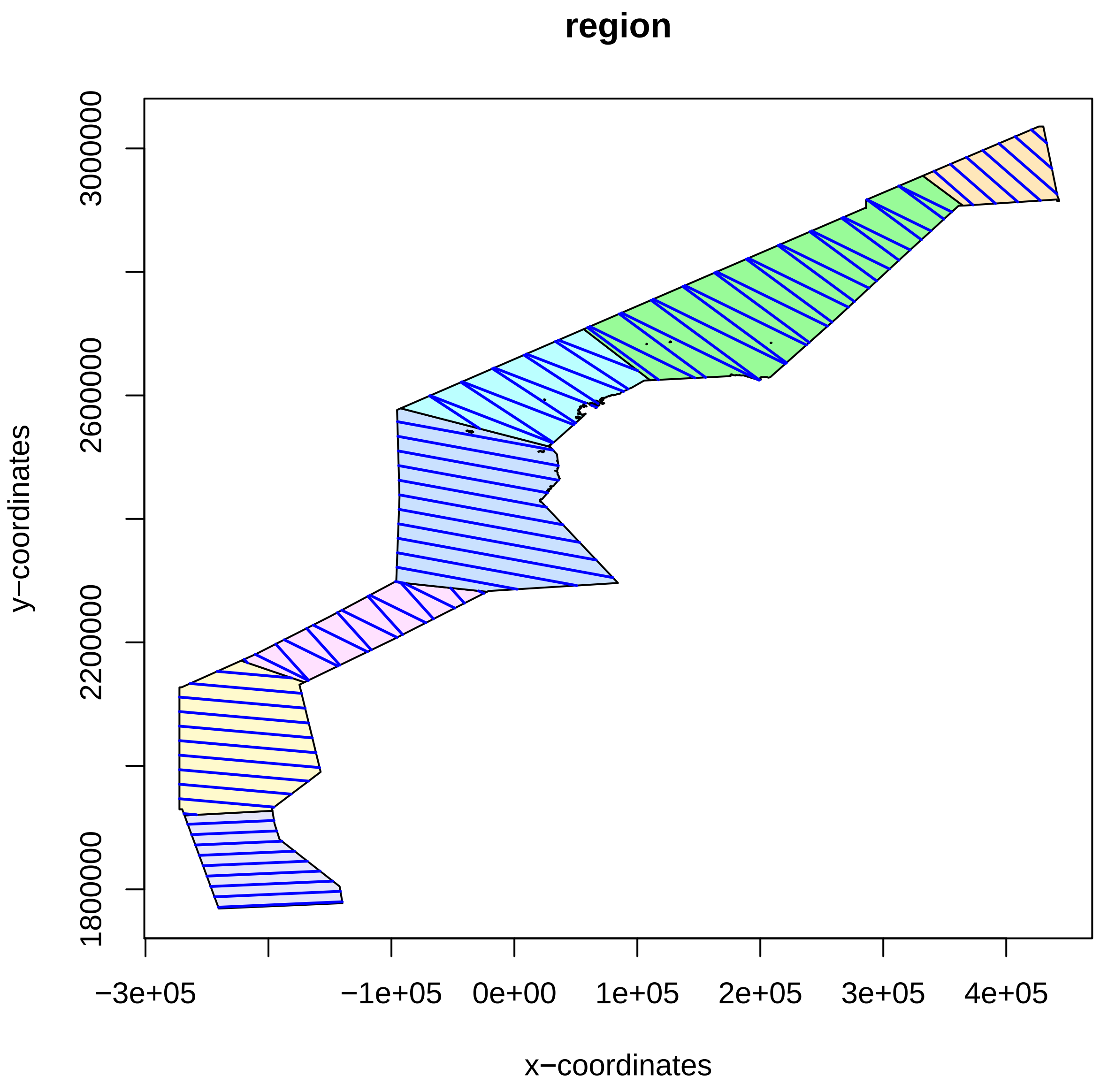
```r
region <- make.region("region", shape = "/Users/lhm/Documents/GitHub/dsuds/dsuds/data/shape/strata.shp"

#Testing out some candidate colours - thoughts?
region.colours <- c("lavender","lemonchiffon", "thistle1", "lightsteelblue1",
                    "paleturquoise1", "palegreen", "wheat1")
plot(region, col = region.colours)
```



This design demonstrates how different strata can have different designs and different design parameters. All designs must either be line transects or point transects, lines and point cannot be mixed. This example uses a mixture of systematic parallel lines and equal spaced zigzags. The zigzags are all generated based on a set spacing where as the parallel lines are generated on a mixture of set spacings or desired number of samplers.

```r
design.mix <- make.design(region,
                          transect.type = "line",
                          design = c(rep("systematic", 2), "eszigzag","systematic",
                                     rep("eszigzag",2), "systematic"),
                          spacing = c(17000, NA, 17000, NA, 17000, 17000, 17000),
                          no.samplers = c(NA, 11, NA, 12, NA, NA, NA),
                          design.angle = c(85, 100, 145, 110, 135, 140, 150),
                          bounding.shape = c(NA, NA, "rectangle", NA, "rectangle",
                                             "rectangle", NA),
                          edge.protocol = rep("minus", 7),
                          truncation = 0.5)
transects.mix <- generate.transects(design.mix)
plot(region, transects.mix, region.col = region.colours, col = 4, lwd = 1.5)
```



Effort Allocation

In Distance the user can choose to allocate effort to strata based on strata area to try and give even coverage across strata. I plan to implement this by allowing the user to enter a single value for line length (or number of samplers for point transect surveys), they can then optionally specify an effort.allocation which will be a vector of proportions which sum to 1 with a value for each strata. If the user omits the effort.allocation argument then the effort will be allocated according to strata area to try to achieve even coverage.

```r
design.mix <- make.design(region,
                          transect.type = "line",
                          design = c(rep("systematic", 2), "eszigzag","systematic",
                                     rep("eszigzag",2), "systematic"),
                          line.length = 4800000,
                          design.angle = c(85, 100, 145, 110, 135, 140, 150),
                          bounding.shape = c(NA, NA, "rectangle", NA, "rectangle",
                                             "rectangle", NA),
                          edge.protocol = rep("minus", 7),
                          truncation = 0.5)
```

Effort Allocation - advanced

If the user wishes to only share effort between a sub group of strata they may do this as shown in the code below. In this case the user will need to allocate -1 in the effort allocation argument where they wish the effort to be allocated based on stratum area. So here the first 4 strata will have line lenghs 500000,700000,500000,700000 respectively and the last 3 strata will have a line length of 2040000 shared between them based on strata area. Otherwise they can again specify proportions that sum to 1 for the non NA values. Is this necessary - not high on the to-do-list,

```r
design.mix <- make.design(region,
                          transect.type = "line",
                          design = c(rep("systematic", 2), "eszigzag","systematic",
                                     rep("eszigzag",2), "systematic"),
                          line.length = c(500000,700000,500000,700000, 2040000, 2040000, 2040000),
                          effort.allocation = c(NA, NA, NA, NA, -1, -1, -1),
                          design.angle = c(85, 100, 145, 110, 135, 140, 150),
                          bounding.shape = c(NA, NA, "rectangle", NA, "rectangle",
                                             "rectangle", NA),
                          edge.protocol = rep("minus", 7),
                          truncation = 0.5)
```

Coverage Grid

I have a little more debugging to do to get the coverage grid creation working. But it will operate with the same options as in Distance, the user will need to supply the region (can have multiple strata and the function to create will form a union from them) and either a spacing or a desired number of points. The function will be called make.coverage(region, spacing, no.grid.points)