Course Notes Advanced SWAT: extracting the ET calibration

Willem Vervoort

24-07-2017

Introduction

This document demonstrates the code to extract the results from the ET calibration of SWAT. In particular it demonstrates how to plot the individual ET comparison timeseries between predicted and observed from the model results. In addition, it demonstrates code how to plot the final performance of the ET calibration for locations in space across the catchment. This base code can be easily adapted to plot different performance measures or different output.

Extracting the timeseries and plotting

This section shows how you can extract the predicted and observed timeseries from the model output and plot these to compare.

Packages

The first step is to load the requires packages. These packages are kind of standard in most of my analyses, especially when dealing with timeseries:

```
require(zoo)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# and I always load tidyverse as it loads ggplot2
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```r
# in this case I also need hydroGOF as this package has the functions for KGE and NSE
# you might have to install this
require(hydroGOF)
```

```
## Loading required package: hydroGOF
```

Preliminaries

There is again a specialised function that allows you to extract the data from output.sub and actually allows you to extract a specific subbasin. It requires as input the directory where output.sub is located and the subbasin you would like to extract. It then extracts the relevant water balance columns. You could adjust the function to extract different other columns in the subbasin file, such as total N.

```r
source("UruguayCourse/functions/Extract_subbasin.R")
# Demonstrate
test <- extract.sub("Cotter2017.PSO.SwatCup",sb=1)
```

```
## Loading required package: data.table
## --------------------------------------------------------------------------
## data.table + dplyr code now lives in dtplyr.
## Please library(dtplyr)!
## --------------------------------------------------------------------------
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
## The following object is masked from 'package:purrr':
##
##     transpose
##
Read 0.0% of 54775 rows
Read 54775 rows and 1 (of 1) columns from 0.013 GB file in 00:04:41
```

```r
head(test)
```

```
## # A tibble: 6 × 9
##   Subbasin    Area   Pcp    ET   PET   SWc   Surf.Q  GW.Q Water.yield
##      <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl>       <dbl>
## 1        1 0.20379   0.0 2.850  7.07   142 8.89e-07     0      0.0788
## 2        1 0.20379   1.5 3.340  5.99   141 8.89e-07     0      0.0782
## 3        1 0.20379   0.0 2.490  6.46   138 8.89e-07     0      0.0776
## 4        1 0.20379   0.0 1.940  4.78   136 8.89e-07     0      0.0770
## 5        1 0.20379   0.0 0.501  1.22   136 8.89e-07     0      0.0764
## 6        1 0.20379   0.1 1.800  4.17   134 8.89e-07     0      0.0758
```

I have also prepared a file with the ET data from the different subcatchments. This is essentially just the output from the function `MODIS.ts` which we have run before, but I saved the output as an RDS file using `saveRDS()`, so I don't have to rerun the function every time. I can now just read it back in

```r
# read in the observed ET data
ET_obs <- readRDS("UruguayCourse/data/ETData.RDS")
head(ET_obs)
```

```
##   Year JDay   ET Point       Date
## 1 2000    1 29.1     1 2000-01-01
## 2 2000    9 36.9     1 2000-01-09
## 3 2000   17 28.3     1 2000-01-17
## 4 2000   25 25.9     1 2000-01-25
## 5 2000   33 25.9     1 2000-02-02
## 6 2000   41 25.9     1 2000-02-10
```

Calibrated only on flow data.

In this case it is a bit more elaborate as we don't have this straight forward in the output as we have for the other iterations.

It is importnat to remember that you need to move output.sub from the SWAT simulation to the iteration folder to make sure it does not get overwritten by later calibrations. Otherwise you need to rerun the calibration.

The idea is to read in the output from the output.sub file, `zoo()` this and then merge this with the observed data and plot.

But we need to loop through all the subcatchments and I will straight away calculate the performance in each subbasin. That way I can use this later in the spatial plotting.

I am also going to save all the comparisons in a stacked dataframe, as that way I can use ggplot to make a nice plot.

```r
# Create an empty storage frame for the performance statistics
summary_ETstats <- data.frame(sub = 1:25, KGE = rep(0,25),
                              NSE = rep(0,25))
# create an empty list for the results
ET_all <- list()


for (i in 1:25 ){
  ET_sub1 <- extract.sub(paste(getwd(),
                     "Cotter2017.PSO.SwatCup/iterations/flowcalibration",sep="/"),
                          sb=i)

  ET_sub1_z <- zoo(ET_sub1$ET, order.by=seq.Date(as.Date("2006-01-01"),
                                                 as.Date("2011-12-31"),1))

  # subset a point, zoo data and select window
  ET_obs1 <- ET_obs[ET_obs["Point"]==i,]
  ET_obs1_z <- zoo(ET_obs1$ET,order.by=ET_obs1$Date)
  ET_obs1_c <- window(ET_obs1_z,start=as.Date("2006-01-01"),end=as.Date("2011-12-31"))

  # merge into a single list
  # Note that all ET observed values are simply divided by 8 days.
  # This introduces an error
  ET_temp <- merge(ET_sub1_z,ET_obs1_c/8,all=F)
  ET_all[[i]] <- data.frame(ET_pred=ET_temp[,1],ET_obs=ET_temp[,2],
```

```
                            Point=rep(i,nrow(ET_temp)))

  summary_ETstats$KGE[i] <- KGE(ET_temp[,1],ET_temp[,2])
  summary_ETstats$NSE[i] <- NSE(ET_temp[,1],ET_temp[,2])
}
```

```
##
Read 0.0% of 54775 rows
Read 54775 rows and 1 (of 1) columns from 0.013 GB file in 00:04:33
```

```
# "unlist" the long list into a stacked dataframe using 'do.call()'
ET_fin <- do.call(rbind, ET_all)
head(ET_fin)
```

```
##            ET_pred ET_obs Point
## 2006-01-01   1.120 5.0375     1
## 2006-01-09   1.620 4.5750     1
## 2006-01-17   4.230 4.4625     1
## 2006-01-25   2.490 4.5625     1
## 2006-02-02   3.810 4.3500     1
## 2006-02-10   0.901 3.7750     1
```

There is one problem with this analysis, I have just divided all the observed data by 8 (which means I make an error on the 1st of january, as this data is only 5 or 4 accumulated days, as day 361 is the last day for 8 day gaps). But we will ignore this for now.

Now make a plot to show each of the subcatchments.

```
p <- ggplot(ET_fin, aes(x = as.Date(row.names(ET_fin)), y = ET_pred)) +
  geom_line(colour="blue") + geom_point(aes(x = as.Date(row.names(ET_fin)),
                                            y = ET_obs), colour="red") +
  facet_wrap(~Point) + xlab("Date")
print(p)
```

Timeseries plot for one of the other calibrations

OK, this clearly demonstrates that there is variation in the calibration results by subcatchments. The question is whether this improves for one of the calibrations. Here I am showing the results of one of my last calibrations. In this case we don't need to calculate the performances as this is already an output from SWAT-CUP, and we also don't need to use output.sub as we can make use of one of the outputfiles from SWAT-CUP: "best_sim.txt".

However in this case we need to specifically define the dates at which the ET is observed.

```
# create an empty list for the results
ET_all <- list()

# sequence of dates
Dates <- c(seq.Date(as.Date("2006-01-01"),
                    as.Date("2006-12-31"),8),
           seq.Date(as.Date("2007-01-01"),
                    as.Date("2007-12-31"),8),
           seq.Date(as.Date("2008-01-01"),
                    as.Date("2008-12-31"),8),
           seq.Date(as.Date("2009-01-01"),
                    as.Date("2009-12-31"),8),
```

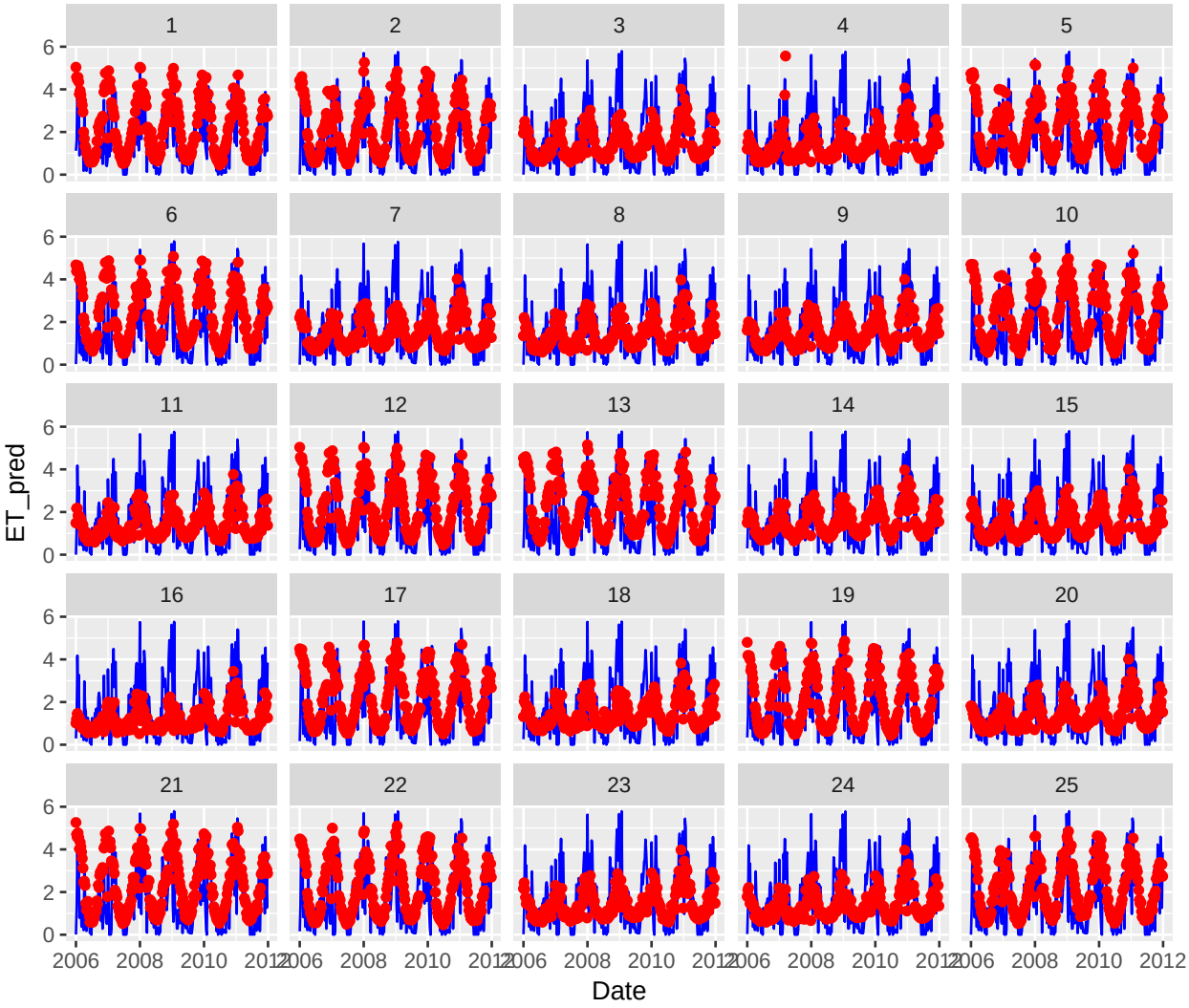

Figure 1: Comparison of predicted and observed ET for the Cotter catchment based on calibration with flow only.

```
           seq.Date(as.Date("2010-01-01"),
                    as.Date("2010-12-31"),8),
           seq.Date(as.Date("2011-01-01"),
                    as.Date("2011-12-31"),8))

# read in the "best_sim.txt" file from PSO.out
foo_bar <- "Cotter2017.PSO.SwatCup/iterations/8thETCalibration/pso.out/best_sim.txt"
foo <- file(foo_bar, "r+")
test <- readLines(foo)


for (i in 1:25) {
  lineno <- grep(paste("ET_",i, sep=""),test)[1]
  ETdata <- fread(foo_bar, data.table=T, skip = lineno, nrows=276, header=T)

  ET_all[[i]] <- data.frame(Dates=Dates, ET_obs= ETdata$observed,
                            ET_pred = ETdata$simulated, Point = rep(i,length(Dates)))
}

# "unlist" the long list into a stacked dataframe using 'do.call()'
ET_fin <- do.call(rbind, ET_all)
```

Again make a plot to show each of the subcatchments.

```
p <- ggplot(ET_fin, aes(x = Dates, y = ET_pred)) +
  geom_line(colour="blue") + geom_point(aes(x = Dates,
                                            y = ET_obs), colour="red") +
  facet_wrap(~Point) + xlab("Date")
print(p)
```

Plotting the performance in space

This section shows how you can extract the performance of the model at each location and plot this in space on a shape of the catchment.

Packages

The first step is to load the required packages for the analysis. In this case the following packages are needed:

```
# make sure all packages are updated
require(raster)
```

```
## Loading required package: raster
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:data.table':
##
##     shift
```

```
## The following object is masked from 'package:dplyr':
```

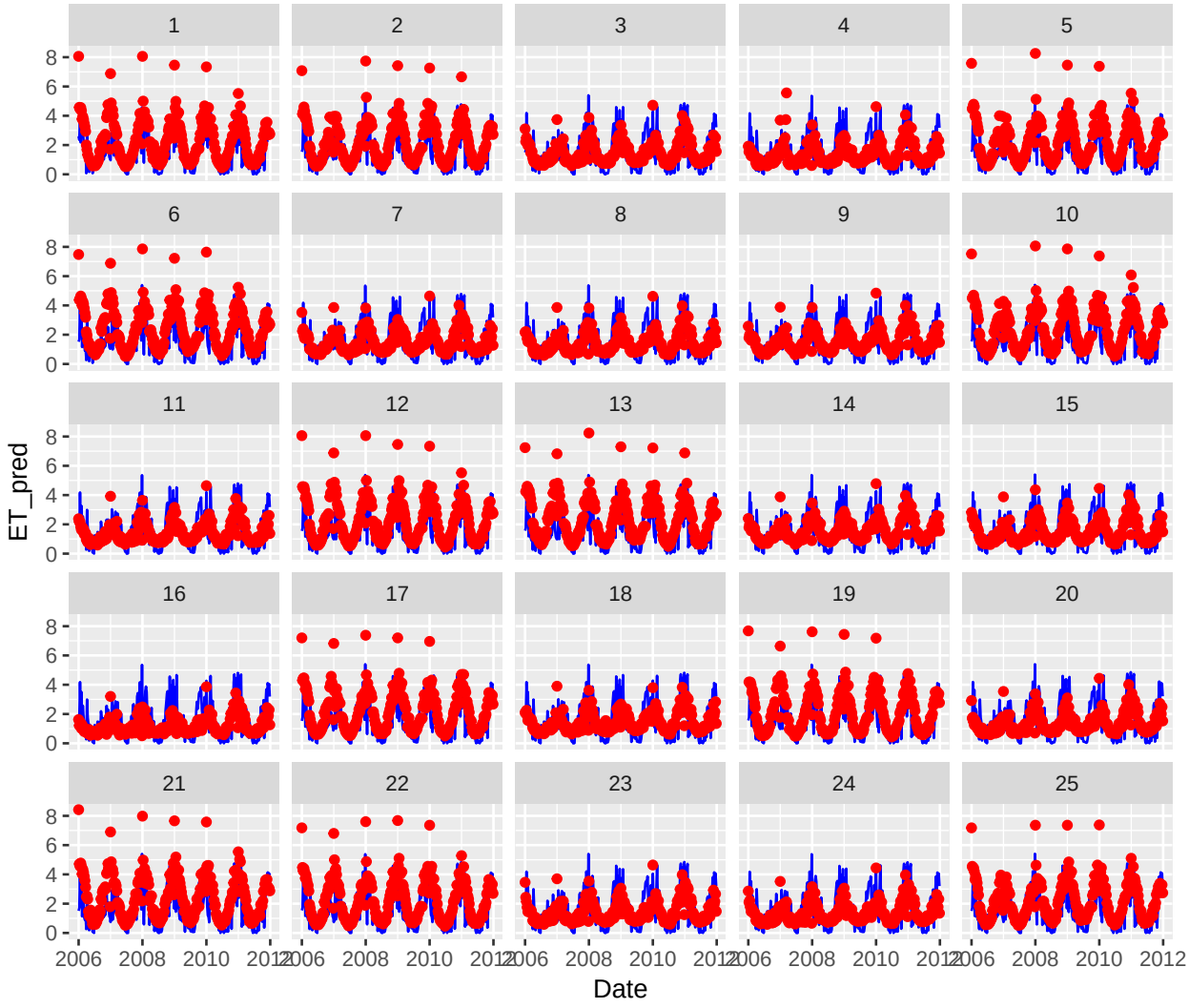

Figure 2: Comparison of predicted and observed ET for the Cotter catchment based on calibration for ET and flow with 10% weigthing on flow, and ET weighted by size of subcatchment.

```
##
##     select
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
require(maptools)
```

```
## Loading required package: maptools
```

```
## Checking rgeos availability: TRUE
```

```
require(rgdal)
```

```
## Loading required package: rgdal
```

```
## rgdal: version: 1.2-6, (SVN revision 651)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/Users/rver4657/Documents/R/win-library/3.3/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/Users/rver4657/Documents/R/win-library/3.3/rgdal/proj
##  Linking to sp version: 1.2-4
```

shapefiles and latitude and longitudes

Also required is a shapefile of the catchment. You probably have this in your GIS files when you developed the SWAT model in QGIS or ARCGIS. Here I am demonstrating the Cotter catchment. After you have loaded the shapefile, you need to make sure it is in the correct projection. This should be OK, but I have put in the code so you can see how to do this.

```
# Reading the shape file of the catchment: move this to the Inputdata dir
cotter <- readShapePoly("Inputdata/CotterShape/CotterLatLong.shp")
```

```
## Warning: use rgdal::readOGR or sf::st_read
```

```
# setting up the projection of the shapefile
proj <- "+proj=longlat +ellps=WGS84"
crs(cotter) <- proj
```

Next read in the latitudes and longitudes of the midpoints of the subbasins, which is where we calibrated on ET.

```
subbasins <- read.csv("InputData/subbasins_cotter.csv")
```

Results from the calibration only on flow

Here we will make use of the summary of the performance which we generated earlier. We can simply call the `summary_ETStats` dataframe. Theperformance results need to be linked to the latitude and longitude results of the calibration

```
sim_res <- summary_ETstats

KGE_sub <- data.frame(Point= subbasins[,1],
                      long = subbasins[,3],
                      lat = subbasins[,2],
                      KGE = sim_res$KGE)
```

```
NSE_sub <- data.frame(Point= subbasins[,1],
                      long = subbasins[,3],
                      lat = subbasins[,2],
                      NSE = sim_res$NSE)


# plotting
gp <- ggplot(cotter, aes(x = long, y = lat)) + geom_polygon(fill="gray75") +
  coord_equal()
```

```
## Regions defined for each Polygons
```

```
gp <- gp + geom_point(data = KGE_sub, aes(x = long, y = lat, col = KGE,
                                          size = KGE)) +
  geom_text(data = KGE_sub, aes(x = long, y = lat,label=Point), vjust=-1)
print(gp)
```

Spatial plot for one of the other calibrations

Again this shows the variation in the calibration results by subcatchment, and that this is not very good. So has this improved for one of the calibrations. Here I am showing the results of one of my last calibrations. and I can just read in "summary_stat.txt" from the PSO.out folder

Again showing the same calibration as the timeseries plots.

```
# now read in the results for eigth ET calibration (changed pars and 0.1 flow)
path <- "Cotter2017.PSO.SwatCup/iterations/8thETCalibration/PSO.OUT/"
sim_res <- read_table(paste(path,"summary_stat.txt",sep=""), skip=3)
```

```
## Parsed with column specification:
## cols(
##   Variable = col_character(),
##   `p-factor` = col_double(),
##   `r-factor` = col_double(),
##   R2 = col_double(),
##   NS = col_double(),
##   bR2 = col_double(),
##   MSE = col_double(),
##   SSQR = col_double(),
##   PBIAS = col_double(),
##   KGE = col_double(),
##   RSR = col_double(),
##   VOL_FR = col_double(),
##   `---` = col_character(),
##   `Mean(sim)` = col_character(),
##   `StdDev(sim)` = col_character()
## )
```

```
# now link results to lat and longs of subbasins
KGE_sub <- data.frame(Point= subbasins[,1],
                      long = subbasins[,3],
                      lat = subbasins[,2],
                      KGE = sim_res$KGE[2:26])
```

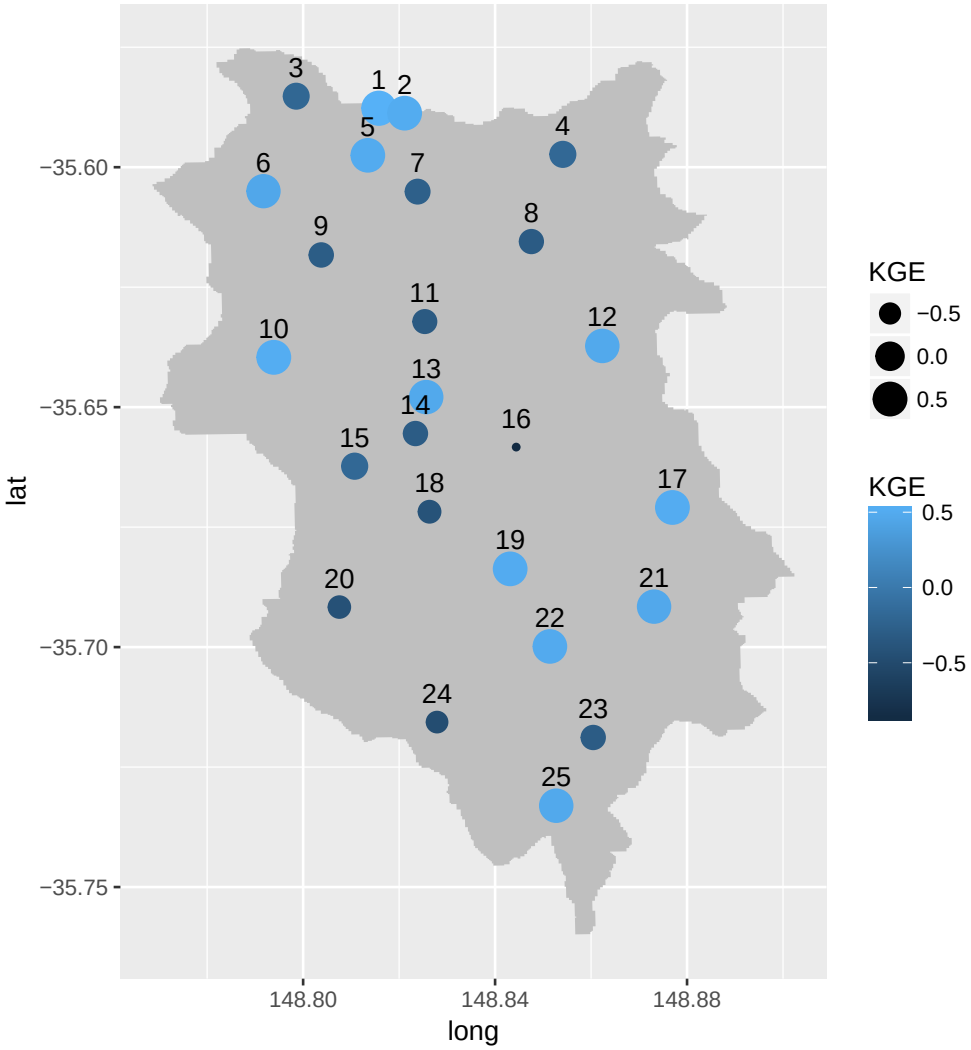

Figure 3: Spatial plot of the performance in the prediction of ET in each subcatchments based on calibration on flow only.

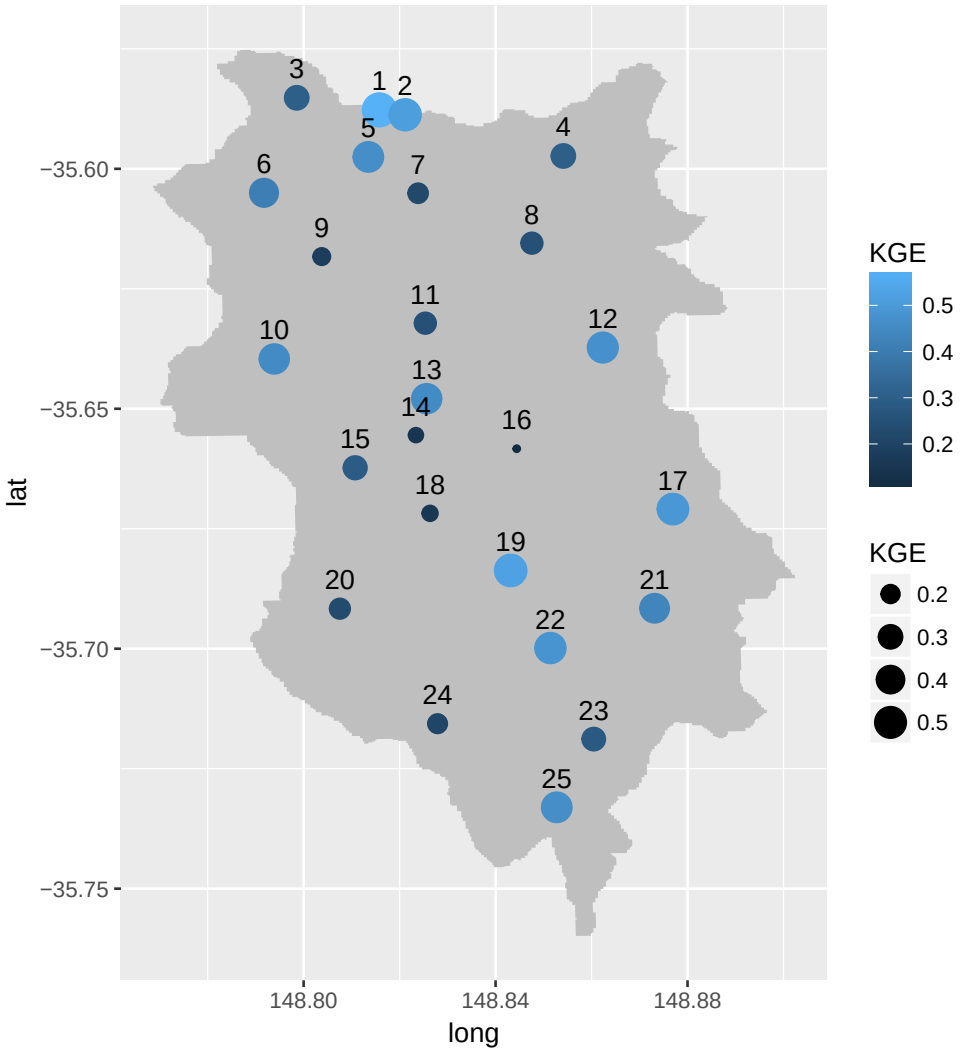

Figure 4: Spatial plot of the performance in the prediction of ET in each subcatchments based on calibration for ET and flow with 10% weigthing on flow, and ET weighted by size of subcatchment.

```
NSE_sub <- data.frame(Point= subbasins[,1],
                      long = subbasins[,3],
                      lat = subbasins[,2],
                      NSE = sim_res$NS[2:26])


# plotting
gp <- ggplot(cotter, aes(x = long, y = lat)) + geom_polygon(fill="gray75") +
  coord_equal()
```

```
## Regions defined for each Polygons
```

```
gp <- gp + geom_point(data = KGE_sub, aes(x = long, y = lat, col = KGE,
                                          size = KGE)) +
  geom_text(data = KGE_sub, aes(x = long, y = lat,label=Point), vjust=-1)
print(gp)
```