Assignment: Regional mapping of climate variables: setting up the dataset

D G Rossiter

2019-10-10

Table of Contents

This script prepares long-term climate records from a set of stations in China for analysis with the methods of the tutorial "Applied geostatistics: Regional mapping of climate variables from point samples", which uses a dataset from the northeastern USA.

Dataset source

This dataset contains 30-year averages of (1) temperature annual average, minimum, maximum, temperature extreme minimum and maximum; (2) precipitation annual average, minimum, maximum.

Data were downloaded from the National Meteorological Information Center [1] as comma-separated (CSV) files by 盛美玲. These files are:

- `temperature_stn_1981_2010.csv`: temperature
- `precip_stn_1981_2010.csv` : precipitation

Preparation

Packages:

```
library(sp)          # spatial data structure for R
library(rgdal)       # coordinate reference systems, interface to geographic
data
library(rgeos)       # manipulate geometry
library(dplyr)       # modern data wrangling
library(stringr)     # modern string manipulation
```

Import of the temperature dataset.

The station and province names use Chinese characters, so we need to identify the encoding. Here are the encoding codes:

```
getOption("encoding")
```

```
## [1] "native.enc"
```

```
# GB-series codes 国家标准
iconvlist()[ix <- grep("GB", iconvlist(), fixed=TRUE)]
```

```
##  [1] "CN-GB"           "CN-GB-ISOIR165"  "CSGB2312"
##  [4] "CSISO57GB1988"   "CSISO58GB231280" "GB_1988-80"
##  [7] "GB_2312-80"      "GB18030"         "GB2312"
## [10] "GBK"             "HZ-GB-2312"
```

```
# UTF-series code: Unicode Transformation Format
iconvlist()[ix <- grep("UTF", iconvlist(), fixed=TRUE)]
```

```
##  [1] "CSUNICODE11UTF7"   "UNICODE-1-1-UTF-7" "UTF-16"
##  [4] "UTF-16BE"          "UTF-16LE"          "UTF-32"
##  [7] "UTF-32BE"          "UTF-32LE"          "UTF-7"
## [10] "UTF-8"             "UTF-8-MAC"         "UTF8"
## [13] "UTF8-MAC"
```

We guess that the encoding is the common `GB18030`; if this does not work, we would try others, until the characters are legible.

---

[1] http://data.cma.cn/site/index.html

Read the temperature dataset; guess the encoding:

```
zhne <- read.csv("./temperature_stn_1981_2010.csv",
                 fileEncoding="GB18030",
                 stringsAsFactors=FALSE)
str(zhne)
```

```
## 'data.frame':    689 obs. of  14 variables:
##  $ station_id                        : int  54398 54399 54406 54416 54419
54419 54421 54424 54431 54431 ...
##  $ average.annual.temperature..      : num  12.3 12.8 9.7 11.3 11.8 12.1
10.4 11.7 13.4 11.7 ...
##  $ Average.annual.maximum.temperature: num  18.1 18.4 16 17.8 18.1 17.5
16.6 17.9 18.4 17.7 ...
##  $ Average.annual.minimum.temperature: num  7.4 7.8 4.3 5.6 6.2 7.2 5.1 6
9 6.2 ...
##  $ Extremely.maximum.temperature     : num  42 41.7 39.2 40.8 41 38.1 40.5
41.3 41.9 38.2 ...
##  $ Extremely.minimum.temperature     : num  -18.4 -20.2 -26.2 -23.3 -22.4
-17.9 -22.3 -22.3 -15.7 -19.1 ...
##  $ province                          : chr  "北京" "北京" "北京" "北京" ...
##  $ station.name                      : chr  "顺义" "海淀" "延庆" "密云" ...
##  $ longitude                         : num  116 116 116 117 116 ...
##  $ latitude                          : num  40.1 39.6 40.3 40.2 40.2 ...
##  $ Altitude.m                        : num  28.6 45.8 487.9 71.8 75.7 ...
##  $ start.year                        : int  198101 198101 198101 198101
198101 198101 198101 198101 198101 198101 ...
##  $ end.year                          : int  201012 201012 201012 201012
201012 201012 201012 201012 201012 201012 ...
##  $ statistical.period                : chr  "198101-201012" "198101-
201012" "200005-201012" "198101-201012" ...
```

Note: we use the R name zhne = 中国东北 because most of these stations are in the northeast of China (although the area is wider than the common definition of 中国东北).

Recode the province names to Latin alphabet, keep the originals:

```
zhne$province <- as.factor(zhne$province)
levels(zhne$province)  # northeast PRC
```

```
## [1] "内蒙古" "北京"   "吉林"   "天津"   "山东"   "山西"   "河北"   "辽宁"
## [9] "黑龙江"
```

```
zhne$province.en <- recode(zhne$province, "北京"="Beijing",  "天津"="Tianjin",
                  "河北"="Hebei",   "山西"="Shanxi",
                  "内蒙古"="NeiMenggu", "辽宁"="Liaoning",
                  "吉林"="Jilin",   "黑龙江"="Heilongjiang",
                  "山东"="Shandong")
table(zhne$province.en)
```

```
##
##    NeiMenggu      Beijing         Jilin      Tianjin     Shandong
##          100           21           53           12          111
##       Shanxi        Hebei     Liaoning Heilongjiang
##          112          150           48           82
```

After some experiments, it became clear that the coördinates are in the format `dd.mm`, not decimal degrees. So we have to convert these to decimal degrees.

```
ddmm.to.dd <- function(ddmm) {
  .deg <- ddmm%/%1
  .min <- (ddmm%%1)*100
  .dd <- .min/60
  return(.deg + .dd)
}
head(zhne$longitude)

## [1] 116.37 116.17 115.58 116.52 116.38 116.38

head(zhne$latitude)

## [1] 40.08 39.59 40.27 40.23 40.22 40.22

zhne$longitude <- ddmm.to.dd(zhne$longitude)
zhne$latitude <- ddmm.to.dd(zhne$latitude)
head(zhne$longitude)

## [1] 116.6167 116.2833 115.9667 116.8667 116.6333 116.6333

head(zhne$latitude)

## [1] 40.13333 39.98333 40.45000 40.38333 40.36667 40.36667
```

Temporary: check xformed coords:

```
tmp <- read.csv("zhne.csv",fileEncoding="GB18030",
                stringsAsFactors=FALSE)
summary(tmp$longitude)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   107.0   115.8   118.9   120.0   124.0   137.0

summary(zhne$longitude)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   106.4   114.4   117.0   118.3   122.3   134.3

summary(tmp$latitude)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   35.14   38.93   41.32   42.03   44.46   57.02

summary(zhne$latitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34.57   37.22   39.65   40.32   42.50   52.97
```

Convert to a `SpatialPointsDataFrame` by specifying the coördinates, keep them also as fields in the data frame.

```
names(zhne)
```

```
##  [1] "station_id"
##  [2] "average.annual.temperature.."
##  [3] "Average.annual.maximum.temperature"
##  [4] "Average.annual.minimum.temperature"
##  [5] "Extremely.maximum.temperature"
##  [6] "Extremely.minimum.temperature"
##  [7] "province"
##  [8] "station.name"
##  [9] "longitude"
## [10] "latitude"
## [11] "Altitude.m"
## [12] "start.year"
## [13] "end.year"
## [14] "statistical.period"
## [15] "province.en"
```

```
coordinates(zhne) <- c("longitude", "latitude")
# they were removed as fields, add them back
zhne$long <- coordinates(zhne)[,1]; zhne$lat <- coordinates(zhne)[,2]
str(zhne)
```

```
## Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
##   ..@ data       :'data.frame':  689 obs. of  15 variables:
##   .. ..$ station_id                       : int [1:689] 54398 54399 54406
54416 54419 54419 54421 54424 54431 54431 ...
##   .. ..$ average.annual.temperature..     : num [1:689] 12.3 12.8 9.7
11.3 11.8 12.1 10.4 11.7 13.4 11.7 ...
##   .. ..$ Average.annual.maximum.temperature: num [1:689] 18.1 18.4 16 17.8
18.1 17.5 16.6 17.9 18.4 17.7 ...
##   .. ..$ Average.annual.minimum.temperature: num [1:689] 7.4 7.8 4.3 5.6
6.2 7.2 5.1 6 9 6.2 ...
##   .. ..$ Extremely.maximum.temperature    : num [1:689] 42 41.7 39.2 40.8
41 38.1 40.5 41.3 41.9 38.2 ...
##   .. ..$ Extremely.minimum.temperature    : num [1:689] -18.4 -20.2 -26.2
-23.3 -22.4 -17.9 -22.3 -22.3 -15.7 -19.1 ...
##   .. ..$ province                         : Factor w/ 9 levels
"内蒙古","北京",..: 2 2 2 2 2 2 2 2 2 2 ...
##   .. ..$ station.name                     : chr [1:689] "顺义" "海淀"
"延庆" "密云" ...
##   .. ..$ Altitude.m                       : num [1:689] 28.6 45.8 487.9
71.8 75.7 ...
##   .. ..$ start.year                       : int [1:689] 198101 198101
```

```
198101 198101 198101 198101 198101 198101 198101 198101 ...
##   .. ..$ end.year                          : int [1:689] 201012 201012
201012 201012 201012 201012 201012 201012 201012 201012 ...
##   .. ..$ statistical.period                : chr [1:689] "198101-201012"
"198101-201012" "200005-201012" "198101-201012" ...
##   .. ..$ province.en                       : Factor w/ 9 levels
"NeiMenggu","Beijing",..: 2 2 2 2 2 2 2 2 2 2 ...
##   .. ..$ long                              : num [1:689] 117 116 116 117
117 ...
##   .. ..$ lat                               : num [1:689] 40.1 40 40.5 40.4
40.4 ...
##   ..@ coords.nrs : int [1:2] 9 10
##   ..@ coords     : num [1:689, 1:2] 117 116 116 117 117 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "longitude" "latitude"
##   ..@ bbox       : num [1:2, 1:2] 106.4 34.6 134.3 53
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "longitude" "latitude"
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr NA
```

Find the time period for each station's records:

```
table(zhne$start.year)

##
## 198101 198112 198601 199101 199201
##    684      1      1      2      1

table(zhne$end.year)

##
## 201012
##    689
```

Use just the stations with the full 30-year record:

```
(n <- length(ix <- which(zhne$start.year != 198101)))

## [1] 5

if (n > 0) { zhne <- zhne[-ix,] }
dim(zhne)

## [1] 684  15
```

Remove the date columns:

```
c.ix <- which(!is.na(match(names(zhne), "start.year"))); zhne <- zhne[, -
c.ix]
```

```
c.ix <- which(!is.na(match(names(zhne), "end.year"))); zhne <- zhne[, -c.ix]
c.ix <- which(!is.na(match(names(zhne), "statistical.period"))); zhne <-
zhne[, -c.ix]
summary(zhne)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                 min       max
## longitude 106.40000 134.28333
## latitude   34.56667  52.96667
## Is projected: NA
## proj4string : [NA]
## Number of points: 684
## Data attributes:
##    station_id    average.annual.temperature..
##  Min.   :50136   Min.   :-4.00
##  1st Qu.:53584   1st Qu.: 5.60
##  Median :54148   Median : 9.60
##  Mean   :53769   Mean   : 8.93
##  3rd Qu.:54596   3rd Qu.:12.80
##  Max.   :58032   Max.   :14.90
##
##  Average.annual.maximum.temperature Average.annual.minimum.temperature
##  Min.   : 0.40                      Min.   :-12.100
##  1st Qu.:12.10                      1st Qu.: -0.200
##  Median :16.10                      Median :  4.150
##  Mean   :15.14                      Mean   :  3.532
##  3rd Qu.:18.80                      3rd Qu.:  7.800
##  Max.   :20.50                      Max.   : 10.800
##
##  Extremely.maximum.temperature Extremely.minimum.temperature    province
##  Min.   :19.90                 Min.   :-49.60                河北   :150
##  1st Qu.:37.90                 1st Qu.:-34.33                山西   :112
##  Median :39.85                 Median :-25.60                山东   :111
##  Mean   :39.56                 Mean   :-27.27                内蒙古 : 99
##  3rd Qu.:41.30                 3rd Qu.:-20.10                黑龙江 : 81
##  Max.   :44.40                 Max.   :-11.70                吉林   : 52
##                                                              (Other): 79
##  station.name         Altitude.m          province.en       long
##  Length:684         Min.   :   1.3   Hebei       :150   Min.   :106.4
##  Class :character   1st Qu.:  44.6   Shanxi      :112   1st Qu.:114.4
##  Mode  :character   Median : 180.6   Shandong    :111   Median :117.0
##                     Mean   : 392.1   NeiMenggu   : 99   Mean   :118.3
##                     3rd Qu.: 724.2   Heilongjiang: 81   3rd Qu.:122.2
##                     Max.   :2208.3   Jilin       : 52   Max.   :134.3
##                                      (Other)     : 79
##       lat
##  Min.   :34.57
##  1st Qu.:37.20
```

```
##  Median :39.64
##  Mean   :40.30
##  3rd Qu.:42.44
##  Max.   :52.97
##
```

Shorten the names of the data fields, for easier typing in R models:

```
names(zhne)
```

```
##  [1] "station_id"
##  [2] "average.annual.temperature.."
##  [3] "Average.annual.maximum.temperature"
##  [4] "Average.annual.minimum.temperature"
##  [5] "Extremely.maximum.temperature"
##  [6] "Extremely.minimum.temperature"
##  [7] "province"
##  [8] "station.name"
##  [9] "Altitude.m"
## [10] "province.en"
## [11] "long"
## [12] "lat"
```

```
names(zhne)[1] <- "station.id"
names(zhne)[2:6] <- c("t.avg", "t.avg.max", "t.avg.min", "t.ext.max",
"t.ext.min")
names(zhne)[9] <- "elev.m"
names(zhne)
```

```
##  [1] "station.id"   "t.avg"        "t.avg.max"    "t.avg.min"
##  [5] "t.ext.max"    "t.ext.min"    "province"     "station.name"
##  [9] "elev.m"       "province.en"  "long"         "lat"
```

Assign the Coordinate Reference System (CRS) to the spatial dataset. Clearly, the coordinates are long/lat:

```
bbox(zhne)
```

```
##                 min       max
## longitude 106.40000 134.28333
## latitude   34.56667  52.96667
```

Since this is older data, we do not think this is WGS84 for the long/lat.

盛美玲 could not find the information about what datum/ellipse was used for the lon/lat in these datasets. We suspect that the Krasovsky 1942 ellipse is correct. Find it in the EPSG database and set the CRS accordingly,

```
getPROJ4VersionInfo()
```

```
## [1] "Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]"
```

```
str(ellps <- projInfo("ellps"))
```

```
## 'data.frame':    43 obs. of  4 variables:
##  $ name       : Factor w/ 43 levels "airy","andrae",..: 30 37 22 25 1 3 34
31 2 4 ...
##  $ major      : Factor w/ 40 levels "a=6370997.0",..: 19 17 19 22 15 20 24
12 6 29 ...
##  $ ell        : Factor w/ 26 levels "b=6355834.8467",..: 17 17 18 17 4 15
15 3 23 15 ...
##  $ description: Factor w/ 43 levels "Airy 1830","Andrae 1876 (Den.,
Iclnd.)",..: 30 38 20 24 1 3 33 31 2 4 ...
```

```
(ix <- grep("Krassovsky, 1942", ellps$description))
```

```
## [1] 31
```

```
ellps[ix,]  ## this is the one we want
```

```
##     name       major      ell      description
## 31 krass a=6378245.0 rf=298.3 Krassovsky, 1942
```

```
str(proj <- projInfo("proj"))
```

```
## 'data.frame':    143 obs. of  2 variables:
##  $ name       : Factor w/ 143 levels "aea","aeqd","airy",..: 1 2 3 4 5 6 7
8 9 10 ...
##  $ description: Factor w/ 141 levels "Airy","Aitoff",..: 3 6 1 2 73 4 5 7
8 9 ...
```

```
(ix <- grep("Lat/long", proj$description))
```

```
## [1] 58 59 60 61
```

```
proj[ix,]  # all aliases
```

```
##       name               description
## 58  lonlat       Lat/long (Geodetic)
## 59  latlon Lat/long (Geodetic alias)
## 60 latlong Lat/long (Geodetic alias)
## 61 longlat Lat/long (Geodetic alias)
```

```
# if we think it is Krassovsky
(proj4string(zhne) <- "+proj=longlat +ellps=krass")
```

```
## [1] "+proj=longlat +ellps=krass"
```

```
# if we think it is WGS84
# (proj4string(zhne) <- "+init=epsg:4326")
```

Import of the precipitation dataset

Read from the CSV file:

```
ds.p <- read.csv("./precip_stn_1981_2010.csv",
                 fileEncoding="GB18030",
                 stringsAsFactors=FALSE)
str(ds.p)
```

```
## 'data.frame':    689 obs. of  12 variables:
##  $ station_id                    : int  54398 54399 54406 54416 54419
54419 54421 54424 54431 54431 ...
##  $ Average.annual.precipitation.mm: num  572 558 424 628 585 ...
##  $ the.maximum.precipitation     : num  866 936 550 924 948 ...
##  $ the.minimum.precipitation     : num  300 358 296 377 409 ...
##  $ province                      : chr  "北京" "北京" "北京" "北京" ...
##  $ station.name                  : chr  "顺义" "海淀" "延庆" "密云" ...
##  $ longitude                     : num  116 116 116 117 116 ...
##  $ latitude                      : num  40.1 39.6 40.3 40.2 40.2 ...
##  $ Altitude.m                    : num  28.6 45.8 487.9 71.8 75.7 ...
##  $ start.year                    : int  198101 198101 198101 198101
198101 198101 198101 198101 198101 198101 ...
##  $ end.year                      : int  201012 201012 201012 201012
201012 201012 201012 201012 201012 201012 ...
##  $ statistical.period            : chr  "198101-201012" "198101-201012"
"200005-201012" "198101-201012" ...
```

```
names(ds.p)[1] <- "station.id"
```

Limit to the records with a full 30-year series:

```
n <- length(ix <- which(ds.p$start.year != 198101))
if (n > 0) { ds.p <- ds.p[-ix,] }
dim(ds.p)
```

```
## [1] 684  12
```

Add precipitation data to the temperature dataframe

Do the two datasets have the same stations?

```
table(zhne$station.id==ds.p$station.id)
```

```
##
## TRUE
##  684
```

```
table(zhne$station.name==ds.p$station.name)
```

```
##
## TRUE
##  684
```

Yes, so we can add the precipitation records to the temperature dataset, to have a single "climate" set.

```
names(ds.p)
```

```
##  [1] "station.id"                        "Average.annual.precipitation.mm"
##  [3] "the.maximum.precipitation"         "the.minimum.precipitation"
##  [5] "province"                          "station.name"
##  [7] "longitude"                         "latitude"
##  [9] "Altitude.m"                        "start.year"
## [11] "end.year"                          "statistical.period"
```

```
# note there are no 'extremes' here, as in the temperature dataset
names(ds.p)[2:4] <- c("p.avg", "p.avg.max", "p.avg.min")
dim(zhne@data)
```

```
## [1] 684  12
```

```
zhne@data <- cbind(zhne@data, ds.p[,2:4])
dim(zhne)
```

```
## [1] 684  15
```

```
names(zhne)
```

```
##  [1] "station.id"   "t.avg"        "t.avg.max"    "t.avg.min"
##  [5] "t.ext.max"    "t.ext.min"    "province"     "station.name"
##  [9] "elev.m"       "province.en"  "long"         "lat"
## [13] "p.avg"        "p.avg.max"    "p.avg.min"
```

Define a metric CRS and transform to this

To build geostatistical models we need a metric CRS, i.e., with more or less true distances, which is not the case with long/lat.

Define a metric CRS, for this approx. square area, we select Lambert Azimuthal Equal Area (see Snyder references for a discussion of the properties of various projections).

First, look for Chinese CRS defined in the EPSG database[2]:

```
epsg <- make_EPSG()
names(epsg)
```

```
## [1] "code" "note" "prj4"
```

```
epsg[grep("China", epsg$note), 1:2]
```

```
##      code                                    note
## 254  4490 # China Geodetic Coordinate System 2000
## 1905 3415    # WGS 72BE / South China Sea Lambert
## 5257 4479 # China Geodetic Coordinate System 2000
```

[2] https://www.epsg-registry.org

These are too far south, and none are Lambert.

Second, look for Lambert Azimuthal Equal Area projections:

```
epsg[grep("laea", epsg$prj4),1:2]
```

```
##      code                                             note
## 662  2163                 # US National Atlas Equal Area
## 1525 3035                         # ETRS89 / LAEA Europe
## 1898 3408                       # NSIDC EASE-Grid North
## 1899 3409                       # NSIDC EASE-Grid South
## 2061 3571         # WGS 84 / North Pole LAEA Bering Sea
## 2062 3572             # WGS 84 / North Pole LAEA Alaska
## 2063 3573             # WGS 84 / North Pole LAEA Canada
## 2064 3574           # WGS 84 / North Pole LAEA Atlantic
## 2065 3575             # WGS 84 / North Pole LAEA Europe
## 2066 3576             # WGS 84 / North Pole LAEA Russia
## 2360 3973 # WGS 84 / NSIDC EASE-Grid North (deprecated)
## 2361 3974 # WGS 84 / NSIDC EASE-Grid South (deprecated)
## 2821 5633                         # PTRA08 / LAEA Europe
## 2823 5635                       # REGCAN95 / LAEA Europe
## 2824 5636                          # TUREF / LAEA Europe
## 2826 5638                       # ISN2004 / LAEA Europe
## 3445 6931          # WGS 84 / NSIDC EASE-Grid 2.0 North
## 3446 6932          # WGS 84 / NSIDC EASE-Grid 2.0 South
```

None for China or even Asia. So we define a custom CRS.

The Lambert Azimuthal Equal Area projection is defined as follows:

- +proj=laea : name of the projection in proj4
- +lon_0= : Longitude of projection center. Defaults to 0.0.
- +lat_0= : Latitude of projection center. Defaults to 0.0.
- +ellps= : See proj -le for a list of available ellipsoids, Defaults to “GRS80”.
- +R= : Radius of the sphere given in meters. If used in conjunction with +ellps +R takes precedence.
- +x_0= : False easting from projection center. Defaults to 0.0.
- +y_0= :False northing from projection center. Defaults to 0.0

Select an ellipsoid

These are possible ellipsoids:

```
sort(as.character(projInfo("ellps")$name))
```

```
##  [1] "airy"      "andrae"    "APL4.9"    "aust_SA"   "bess_nam"
##  [6] "bessel"    "clrk66"    "clrk80"    "clrk80ign" "CPM"
## [11] "delmbr"    "engelis"   "evrst30"   "evrst48"   "evrst56"
## [16] "evrst69"   "evrstSS"   "fschr60"   "fschr60m"  "fschr68"
## [21] "GRS67"     "GRS80"     "helmert"   "hough"     "IAU76"
```

```
## [26] "intl"       "kaula"      "krass"      "lerch"      "MERIT"
## [31] "mod_airy"   "mprts"      "new_intl"   "NWL9D"      "plessis"
## [36] "SEasia"     "SGS85"      "sphere"     "walbeck"    "WGS60"
## [41] "WGS66"      "WGS72"      "WGS84"
```

We select `WGS84`.

Find an appropriate projection centre

We use the centre of the bounding box as the projection centre:

```
bbox(zhne)

##                  min       max
## longitude 106.40000 134.28333
## latitude   34.56667  52.96667

(lon.0 <- round(mean(bbox(zhne)[1,]),1))

## [1] 120.3

(lat.0 <- round(mean(bbox(zhne)[2,]),1))

## [1] 43.8
```

proj4 string for the coördinate reference system

No reason to use false N or E.

```
(zh.crs <- paste0("+proj=laea +ellps=WGS84 +lon_0=", lon.0, " +lat_0=",
lat.0))

## [1] "+proj=laea +ellps=WGS84 +lon_0=120.3 +lat_0=43.8"
```

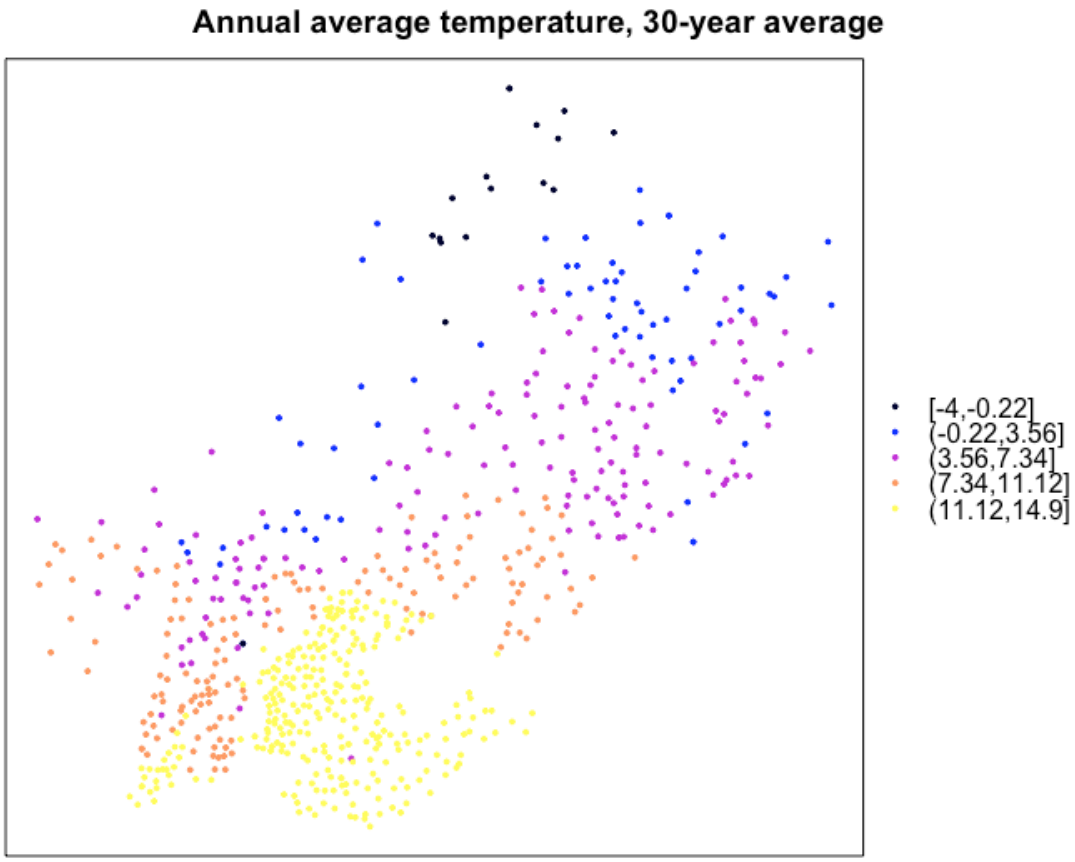
Transform the point coverages

Change the coördinate names to show that this is a metric, not geographic system:

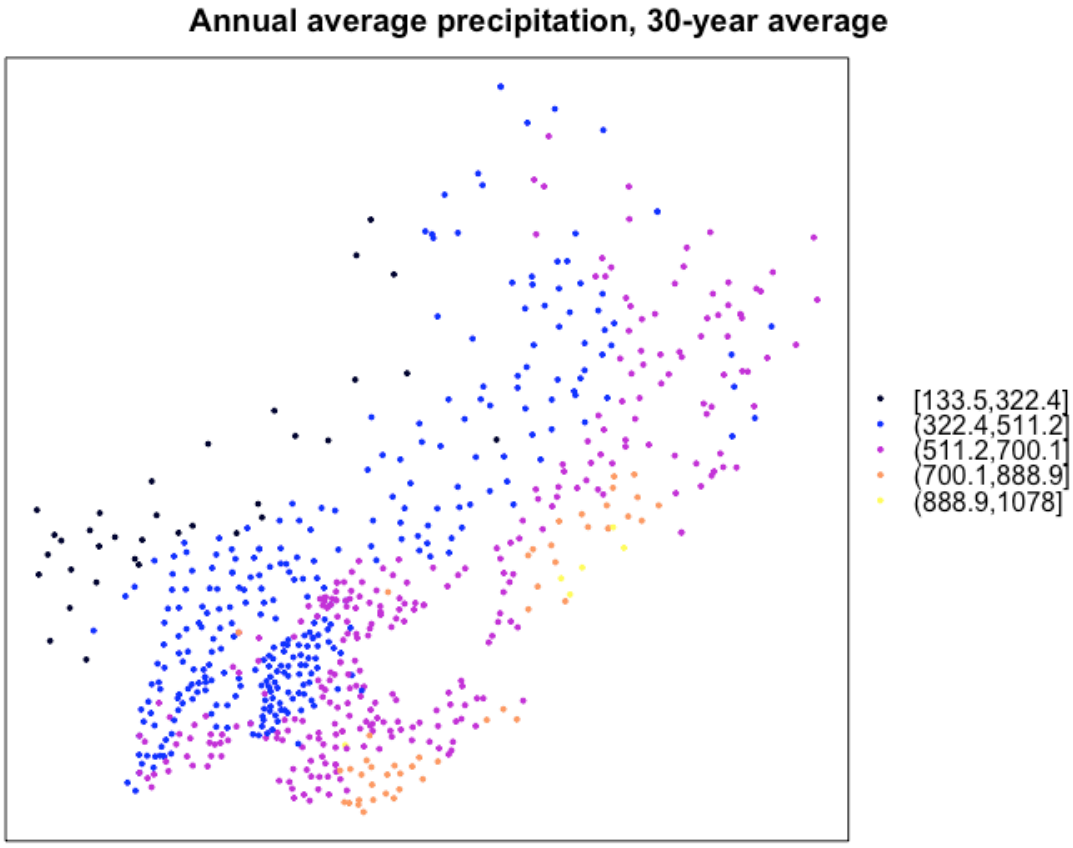
```
zhne.m <- spTransform(zhne, zh.crs)
bbox(zhne.m)

##                min     max
## longitude -1155822 1040008
## latitude  -1020568 1020209

coordnames(zhne.m) <- c("E", "N")
```

Show the points with average annual temperature and precipitation:

```
spplot(zhne.m, zcol="t.avg", key.space="right", pch=20, cex=0.5,
       main="Annual average temperature, 30-year average")
```




```
spplot(zhne.m, zcol="p.avg", key.space="right", pch=20, cex=0.5,
       main="Annual average precipitation, 30-year average")
```



Also make a dataframe version, with the transformed coördinates as data fields, some modelling approaches only use data fields.

```
zhne.df <- as.data.frame(zhne.m)
names(zhne.df)
```

```
##  [1] "station.id"   "t.avg"        "t.avg.max"    "t.avg.min"
##  [5] "t.ext.max"    "t.ext.min"    "province"     "station.name"
##  [9] "elev.m"       "province.en"  "long"         "lat"
## [13] "p.avg"        "p.avg.max"    "p.avg.min"    "E"
## [17] "N"
```

Export point coverages

Export this dataset as a geopackage, to check the locations within a GIS. For this, use the untransformed points. Note that geopackages can contain multiple coverages.

```
writeOGR(zhne, dsn="./climate.gpkg", layer="climate", driver="GPKG",
         overwrite_layer=TRUE)
```

Difficult to check, because no reliable background map.

Political boundaries

The 2nd-level administrative boundaries were downloaded from the Global Administrative Units (GADM) database [3]. Several formats are available (Shapefile, geopackage) as well the R datasets `.rds` in either `sp` or `sf` (Simple Features) formats. However, as of 26-Sep-2019 these seem to be corrupted.

The geopackage has multiple layers, one for each administrative level.

```
ogrInfo(dsn="./gadm36_CHN.gpkg")
```

```
## Source:
"/Users/rossiter/data/edu/dgeostats/ex/ds/中国天气/gadm36_CHN.gpkg", layer:
"gadm36_CHN_0"
## Driver: GPKG; number of rows: 1
## Feature type: wkbPolygon with 2 dimensions
## Extent: (73.5577 18.1593) - (134.774 53.5609)
## CRS: +proj=longlat +datum=WGS84 +no_defs
## Number of fields: 2
##     name type length typeName
## 1  GID_0    4      0   String
## 2 NAME_0    4      0   String
```

```
ogrListLayers(dsn="./gadm36_CHN.gpkg")
```

```
## [1] "gadm36_CHN_0" "gadm36_CHN_1" "gadm36_CHN_2" "gadm36_CHN_3"
## attr(,"driver")
## [1] "GPKG"
## attr(,"nlayers")
## [1] 4
```

Level 1 political units (省级行政区)

Read in the 1st-level units (provinces and equivalent):

```
adm1 <- readOGR(dsn="./gadm36_CHN.gpkg", layer="gadm36_CHN_1")
```

```
## OGR data source with driver: GPKG
## Source:
"/Users/rossiter/data/edu/dgeostats/ex/ds/中国天气/gadm36_CHN.gpkg", layer:
"gadm36_CHN_1"
```

---

[3] https://gadm.org/download_country_v3.html

```
## with 31 features
## It has 10 fields

names(adm1)

##  [1] "GID_0"     "NAME_0"    "GID_1"     "NAME_1"    "VARNAME_1"
##  [6] "NL_NAME_1" "TYPE_1"    "ENGTYPE_1" "CC_1"      "HASC_1"

unique(adm1$NL_NAME_1)  # local name

##  [1] 安徽|安徽                          北京|北京
##  [3] 重慶|重庆                          福建
##  [5] 甘肅|甘肃                          廣東|广东
##  [7] 廣西壯族自治區|广西壮族自治区      貴州|贵州
##  [9] 海南                               河北
## [11] 黑龙江省|黑龍江省                  河南
## [13] 湖北                               湖南
## [15] 江蘇|江苏                         江西
## [17] 吉林                               遼寧|辽宁
## [19] 內蒙古自治區|内蒙古自治区          寧夏回族自治區|宁夏回族自治区
## [21] 青海                               陝西|陕西
## [23] 山東|山东                         上海|上海
## [25] 山西                               四川
## [27] 天津|天津                         新疆維吾爾自治區|新疆维吾尔自治区
## [29] 西藏自治區|西藏自治区              雲南|云南
## [31] 浙江
## 31 Levels: 上海|上海 內蒙古自治區|内蒙古自治区 北京|北京 吉林 ...
黑龙江省|黑龍江省

unique(adm1$NAME_1)     # English name

##  [1] Anhui          Beijing        Chongqing      Fujian
##  [5] Gansu          Guangdong      Guangxi        Guizhou
##  [9] Hainan         Hebei          Heilongjiang   Henan
## [13] Hubei          Hunan          Jiangsu        Jiangxi
## [17] Jilin          Liaoning       Nei Mongol     Ningxia Hui
## [21] Qinghai        Shaanxi        Shandong       Shanghai
## [25] Shanxi         Sichuan        Tianjin        Xinjiang Uygur
## [29] Xizang         Yunnan         Zhejiang
## 31 Levels: Anhui Beijing Chongqing Fujian Gansu Guangdong ... Zhejiang
```

The Chinese names have both simplified and traditional characters, if there is a difference, and sometimese even when not (see “北京|北京”).

We only need some of the columns, this dataset is only to show the political divisions as a background map.

```
adm1@data <- adm1@data %>% select("NAME_1", "NL_NAME_1")
```

Subset these to the provinces in the climate dataset.

Find the simplified character version of the administrative names. Note the use of a regular expression to find the characters at the end of the string, this will pass over "|" and anything before it.

```
(.n1 <- stringr::str_extract(adm1$NL_NAME_1, "[:alpha:]+$" ))
```

```
##  [1] "安徽"              "北京"              "重庆"
##  [4] "福建"              "甘肃"              "广东"
##  [7] "广西壮族自治区"    "贵州"              "海南"
## [10] "河北"              "黑龍江省"          "河南"
## [13] "湖北"              "湖南"              "江苏"
## [16] "江西"              "吉林"              "辽宁"
## [19] "内蒙古自治区"      "宁夏回族自治区"    "青海"
## [22] "陕西"              "山东"              "上海"
## [25] "山西"              "四川"              "天津"
## [28] "新疆维吾尔自治区" "西藏自治区"        "云南"
## [31] "浙江"
```

Do they match the names in the climate database?

```
(.n2 <- levels(zhne$province))
```

```
## [1] "内蒙古" "北京"   "吉林"   "天津"   "山东"   "山西"   "河北"   "辽宁"
## [9] "黑龙江"
```

```
setdiff(.n2,.n1)
```

```
## [1] "内蒙古" "黑龙江"
```

They do not agree, the administrative database uses official names (longer, e.g., "内蒙古自治区" instead of "内蒙古").

So we select directly using the administrative names as they are given in that map:

```
provinces.with.stations <- c("内蒙古自治區|内蒙古自治区",
                             "北京|北京",
                             "吉林",
                             "天津|天津",
                             "山東|山东",
                             "山西",
                             "河北",
                             "遼寧|辽宁",
                             "黑龙江省|黑龍江省")
adm1.ne <- adm1[adm1$NL_NAME_1 %in% provinces.with.stations, ]
# drop unused levels
adm1.ne@data$NL_NAME_1 <- factor(adm1.ne@data$NL_NAME_1)
adm1.ne@data$NAME_1 <- factor(adm1.ne@data$NAME_1)
levels(adm1.ne$NL_NAME_1)
```

```
## [1] "内蒙古自治區|内蒙古自治区" "北京|北京"
## [3] "吉林"                      "天津|天津"
## [5] "山東|山东"                 "山西"
## [7] "河北"                      "遼寧|辽宁"
## [9] "黑龙江省|黑龍江省"
```

```
levels(adm1.ne$NAME_1)
```

```
## [1] "Beijing"      "Hebei"        "Heilongjiang" "Jilin"
## [5] "Liaoning"     "Nei Mongol"   "Shandong"     "Shanxi"
## [9] "Tianjin"
```

```
bbox(adm1.ne)
```

```
##        min       max
## x 97.18472 134.77393
## y 34.38634  53.56086
```

Notice that the bounding box is automatically recomputed from the selected polygons.

However, these boundaries are highly-detailed, too much for our purpose of showing the correct location of stations in political divisions. So we simplify, using a tolerance of 1' (about 1.85 km).

```
proj4string(adm1.ne)
```

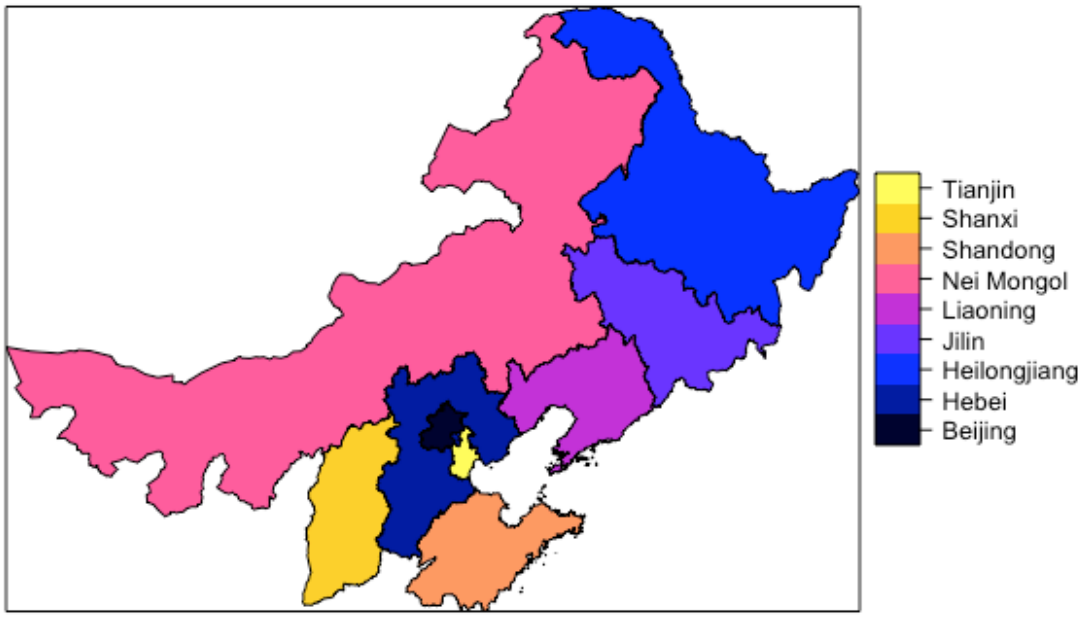
```
## [1] "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

```
adm1.ne.s <- gSimplify(adm1.ne, tol=1/60, topologyPreserve=TRUE)
adm1.ne <- SpatialPolygonsDataFrame(adm1.ne.s, adm1.ne@data)
bbox(adm1.ne)
```

```
##        min       max
## x 97.18472 134.77156
## y 34.39336  53.56004
```

Display a map of these provinces:

```
spplot(adm1.ne, zcol="NAME_1")
```



This simplifies the boundaries but does not remove small polygons within a multipolygon.

Level 2 political units (地级行政区)

Read in the 2nd-level units (counties and equivalent) and keep just relevant fields:

```
adm2 <- readOGR(dsn="./gadm36_CHN.gpkg", layer="gadm36_CHN_2")

## OGR data source with driver: GPKG
## Source:
"/Users/rossiter/data/edu/dgeostats/ex/ds/中国天气/gadm36_CHN.gpkg", layer:
"gadm36_CHN_2"
## with 344 features
## It has 13 fields

names(adm2@data)
```

```
##  [1] "GID_0"     "NAME_0"    "GID_1"     "NAME_1"    "NL_NAME_1"
##  [6] "GID_2"     "NAME_2"    "VARNAME_2" "NL_NAME_2" "TYPE_2"
## [11] "ENGTYPE_2" "CC_2"      "HASC_2"
```

```
adm2@data <- adm2@data %>% select("NAME_1", "NL_NAME_1", "NAME_2",
"NL_NAME_2")
```

List the counties in 江苏:

```
adm2@data %>% filter(NAME_1=="Jiangsu") %>% select(contains("NAME_2"))
```

```
##         NAME_2 NL_NAME_2
## 1    Changzhou    常州市
## 2      Huai'an    淮安市
## 3  Lianyungang  连云港市
## 4      Nanjing    南京市
## 5      Nantong    南通市
## 6       Suqian    宿迁市
## 7       Suzhou    苏州市
## 8      Taizhou    泰州市
## 9         Wuxi    无锡市
## 10      Xuzhou    徐州市
## 11    Yancheng    盐城市
## 12    Yangzhou    扬州市
## 13   Zhenjiang    镇江市
```

Subset to the northeast, simplify the geometry:

```
adm2.ne <- adm2[adm2$NL_NAME_1 %in% provinces.with.stations, ]
adm2.ne@data$NL_NAME_1 <- factor(adm2.ne@data$NL_NAME_1)
adm2.ne@data$NAME_1 <- factor(adm2.ne@data$NAME_1)
adm2.ne@data$NL_NAME_2 <- factor(adm2.ne@data$NL_NAME_2)
adm2.ne@data$NAME_2 <- factor(adm2.ne@data$NAME_2)
levels(adm2.ne$NL_NAME_2)
```

```
##  [1] "七台河市"          "东营市"            "临汾市"
##  [4] "临沂市"            "丹东市"            "乌兰察布市"
##  [7] "乌海市"            "伊春市"            "佳木斯市"
## [10] "保定市"            "兴安盟"            "包头市"
## [13] "北京|北京"         "双鸭山市"          "吉林市"
## [16] "吕梁市"            "呼伦贝尔市"        "呼和浩特市"
## [19] "哈尔滨市"          "唐山市"            "四平市"
## [22] "大兴安岭地区"      "大同市"            "大庆市"
## [25] "大连市"            "天津|天津"         "太原市"
## [28] "威海市"            "巴彦淖尔市"        "廊坊市"
## [31] "延边朝鲜族自治州" "张家口市"          "德州市"
## [34] "忻州市"            "承德市"            "抚顺市"
## [37] "日照市"            "晋中市"            "晋城市"
```

```
## [40] "朔州市"           "朝阳市"           "本溪市"
## [43] "松原市"           "枣庄市"           "沈阳市"
## [46] "沧州市"           "泰安市"           "济南市"
## [49] "济宁市"           "淄博市"           "滨州"
## [52] "潍坊市"           "烟台市"           "牡丹江市"
## [55] "白城市"           "白山市"           "盘锦市"
## [58] "石家庄市"         "秦皇岛市"         "绥化市"
## [61] "聊城市"           "莱芜市"           "菏泽市"
## [64] "葫芦岛市"         "衡水市"           "赤峰市"
## [67] "辽源市"           "辽阳市"           "运城县"
## [70] "通化市"           "通辽市"           "邢台市"
## [73] "邯郸市"           "鄂尔多斯市"       "铁岭市"
## [76] "锡林郭勒盟"       "锦州市"           "长春市"
## [79] "长治市"           "阜新市"           "阳泉市"
## [82] "阿拉善盟"         "青岛市"           "鞍山市"
## [85] "鸡西市"           "鹤岗市"           "黑河市"
## [88] "齐齐哈尔市"
```

```
levels(adm2.ne$NAME_2)
```

```
##  [1] "Alxa"           "Anshan"         "Baicheng"       "Baishan"
##  [5] "Baoding"        "Baotou"         "Baynnur"        "Beijing"
##  [9] "Benxi"          "Binzhou"        "Cangzhou"       "Changchun"
## [13] "Changzhi"       "Chaoyang"       "Chengde"        "Chifeng"
## [17] "Dalian"         "Dandong"        "Daqing"         "Datong"
## [21] "Daxing'anling"  "Dezhou"         "Dongying"       "Fushun"
## [25] "Fuxin"          "Handan"         "Harbin"         "Hegang"
## [29] "Heihe"          "Hengshui"       "Heze"           "Hohhot"
## [33] "Huludao"        "Hulunbuir"      "Jiamusi"        "Jilin"
## [37] "Jinan"          "Jincheng"       "Jining"         "Jinzhong"
## [41] "Jinzhou"        "Jixi"           "Laiwu"          "Langfang"
## [45] "Liaocheng"      "Liaoyang"       "Liaoyuan"       "Linfen"
## [49] "Linyi"          "Luliang"        "Mudanjiang"     "Ordos"
## [53] "Panjin"         "Qingdao"        "Qinhuangdao"    "Qiqihar"
## [57] "Qitaihe"        "Rizhao"         "Shenyang"       "Shijiazhuang"
## [61] "Shuangyashan"   "Shuozhou"       "Siping"         "Songyuan"
## [65] "Suihua"         "Tai'an"         "Taiyuan"        "Tangshan"
## [69] "Tianjin"        "Tieling"        "Tonghua"        "Tongliao"
## [73] "Ulaan Chab"     "Weifang"        "Weihai"         "Wuhai"
## [77] "Xilin Gol"      "Xing'an"        "Xingtai"        "Xinzhou"
## [81] "Yanbian Korean" "Yangquan"       "Yantai"         "Yichun"
## [85] "Yuncheng"       "Zaozhuang"      "Zhangjiakou"    "Zibo"
```

```
summary(adm2.ne)
```

```
## Object of class SpatialPolygonsDataFrame
## Coordinates:
##        min       max
## x 97.18472 134.77393
```

```
## y 34.38634  53.56086
## Is projected: FALSE
## proj4string :
## [+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0]
## Data attributes:
##           NAME_1                         NL_NAME_1       NAME_2
##  Shandong    :17   山東|山东                  :17   Alxa    : 1
##  Heilongjiang:13   遼寧|辽宁                  :13   Anshan  : 1
##  Liaoning    :13   黑龙江省|黑龍江省          :13   Baicheng: 1
##  Nei Mongol  :12   内蒙古自治區|内蒙古自治区:12   Baishan : 1
##  Hebei       :11   山西                       :11   Baoding : 1
##  Shanxi      :11   河北                       :11   Baotou  : 1
##  (Other)     :11   (Other)                    :11   (Other) :82
##       NL_NAME_2
##  七台河市  : 1
##  东营市    : 1
##  临汾市    : 1
##  临沂市    : 1
##  丹东市    : 1
##  乌兰察布市: 1
##  (Other)   :82

adm2.ne.s <- gSimplify(adm2.ne, tol=1/60, topologyPreserve=TRUE)
adm2.ne <- SpatialPolygonsDataFrame(adm2.ne.s, adm2.ne@data)
```

Display the level-2 boundaries inside the provinces:

```
spplot(adm2.ne, zcol="NAME_2", col.regions="white")
```

Make transformed versions to match the metric version of the stations:

```
adm1.ne.m <- spTransform(adm1.ne, zh.crs)
adm2.ne.m <- spTransform(adm2.ne, zh.crs)
bbox(adm1.ne.m)

##        min     max
## x -1866445 1078665
## y -1040802 1087581

bbox(adm2.ne.m)

##        min     max
## x -1866445 1079156
## y -1040802 1087680

bbox(zhne.m)
```

```
##        min     max
## E -1155822 1040008
## N -1020568 1020209
```

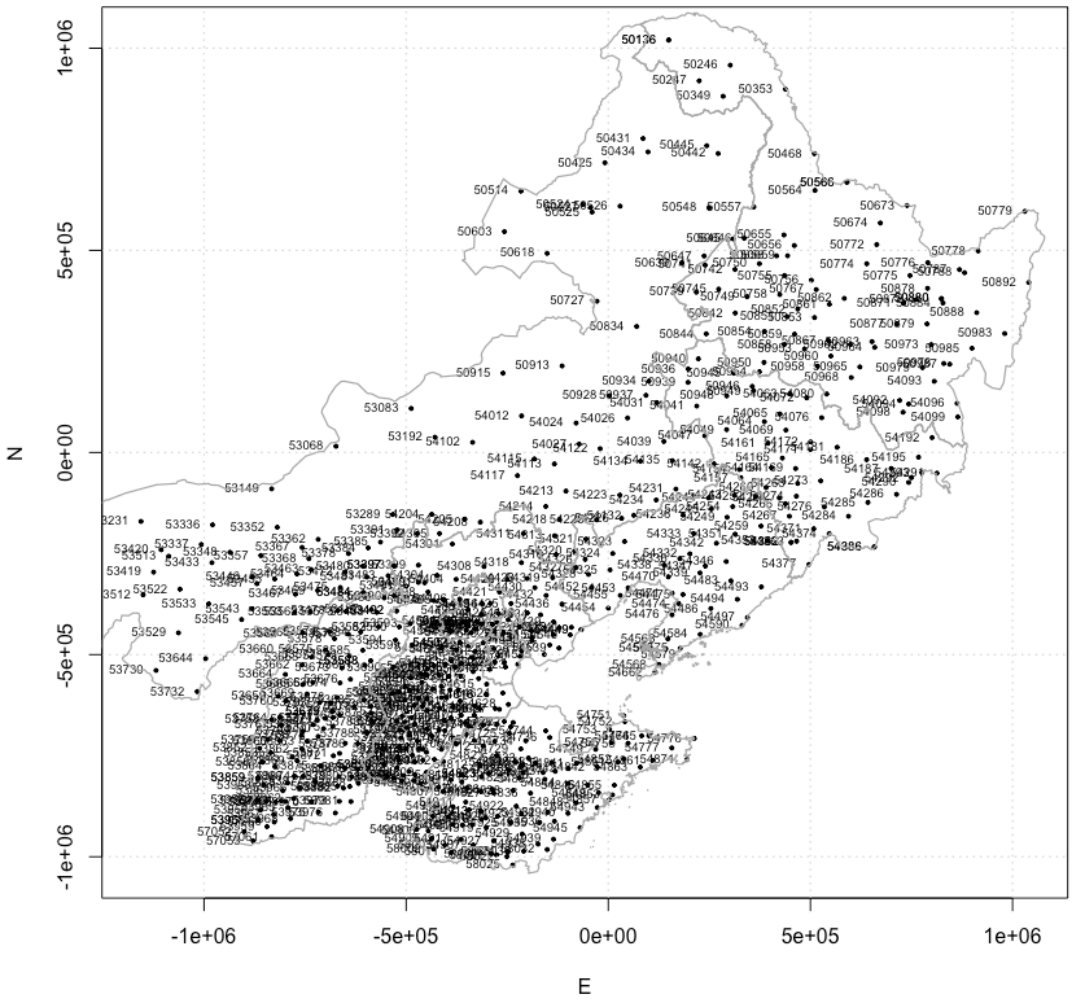
Size of (multi-)polygons at the province level:

```
str(poly.areas <- sapply(slot(adm1.ne.m, "polygons"),
                  function(x) sapply(slot(x, "Polygons"), slot, "area")))
```

```
## List of 9
##  $ : num 1.64e+10
##  $ : num [1:6] 4.34e+06 2.11e+06 4.14e+04 1.70e+05 1.20e+09 ...
##  $ : num 4.55e+11
##  $ : num 1.91e+11
##  $ : num [1:163] 280038 3380158 66240 705301 130438 ...
##  $ : num [1:2] 2.00e+05 1.15e+12
##  $ : num [1:104] 193380 57141 77270 47225 600623 ...
##  $ : num 1.56e+11
##  $ : num 1.17e+10
```

Show stations on political boundaties:

```
.pal <- palette(terrain.colors(9))
plot(coordinates(zhne.m), asp=1, xlab="E", ylab="N", pch=20, cex=0.6)
# ,     col=zhne.m$province)
text(coordinates(zhne.m),
  labels=zhne.m$station.id, cex=0.6,
  pos=2)
plot(adm1.ne.m, add=TRUE, border="darkgray", lwd=1.2)
grid()
```

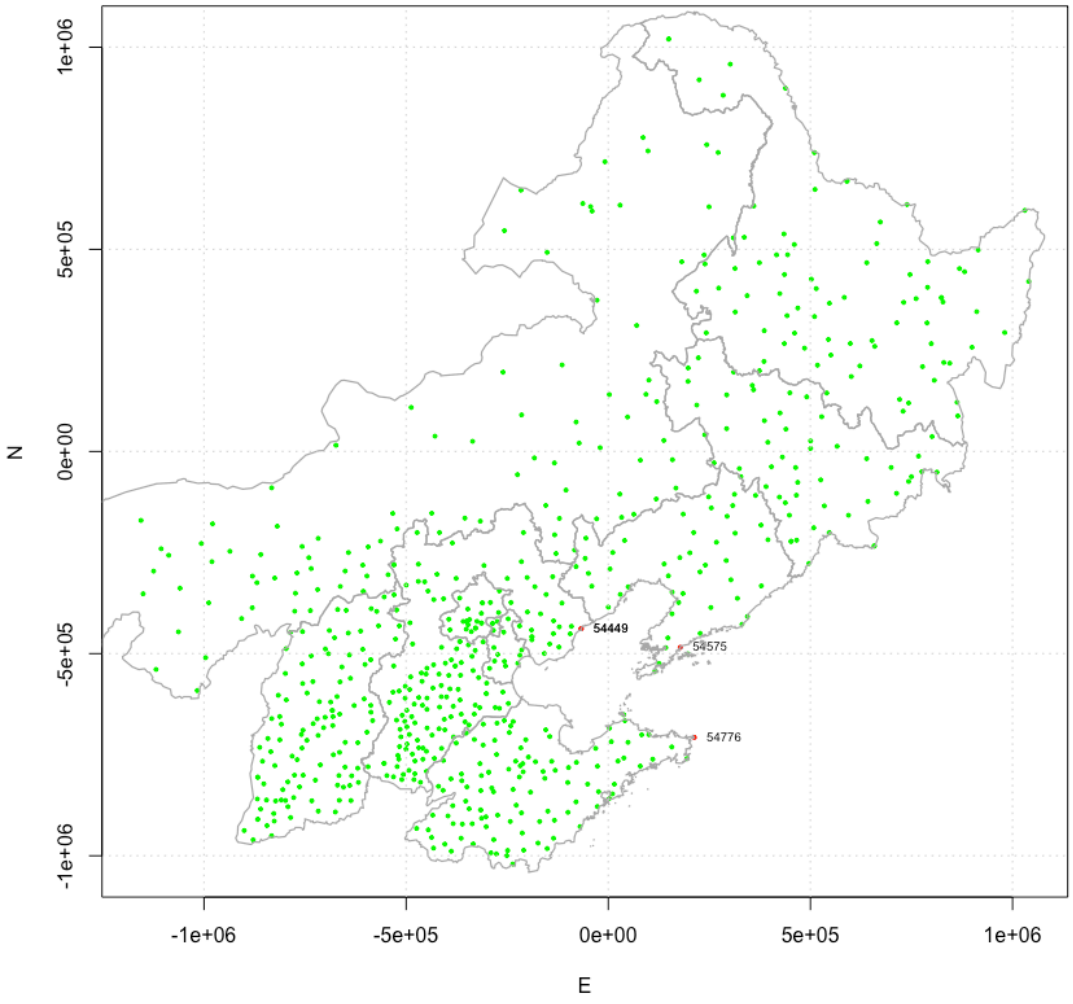


Some stations are not in their declared boundaries:

```
ix <- over(zhne.m, adm1.ne.m)
str(ix)

## 'data.frame':    684 obs. of  2 variables:
##  $ NAME_1   : Factor w/ 9 levels "Beijing","Hebei",..: 1 1 1 1 1 1 1 1 1 1
...
##  $ NL_NAME_1: Factor w/ 9 levels "內蒙古自治區|内蒙古自治区",..: 2 2 2 2 2 2
2 2 2 2 ...

length(no.province <- which(is.na(ix$NAME_1)))

## [1] 4

zhne.m@data[no.province, c("station.id", "province", "station.name", "long",
"lat")]
```

```
##     station.id province station.name     long      lat
## 108      54449     河北       秦皇岛 119.5167 39.85000
## 109      54449     河北       秦皇岛 119.5167 39.85000
## 439      54575     辽宁         皮口 122.3667 39.41667
## 603      54776     山东       成山头 122.7000 37.40000
```

```
plot(coordinates(zhne.m), asp=1, xlab="E", ylab="N", pch=20, cex=0.6,
     col=ifelse(is.na(ix$NAME_1), "red", "green"))
text(coordinates(zhne.m),
  labels=ifelse(is.na(ix$NAME_1), zhne.m$station.id, ""), cex=0.6,
  pos=4)
plot(adm1.ne.m, add=TRUE, border="darkgray", lwd=1.2)
grid()
```



These are islands just off shore, not inside the boundaries from GADM. Locations are good.

Save stations and provinces as R objects

Save as an R object:

```
save(zhne, zhne.m, zhne.df, adm1.ne, adm2.ne, adm1.ne.m, adm2.ne.m, zh.crs,
file="./zhne_stations.RData")
```

References

- Bugayevskiy, L. M., & Snyder, J. P. (1995). Map projections: a reference manual. Taylor & Francis.
- Snyder, J. P. (1987). Map projections: a working manual. Retrieved from https://pubs.er.usgs.gov/publication/pp1395
- Snyder, J. P., & Voxland, P. M. (1989). An album of map projections. Retrieved from https://pubs.er.usgs.gov/publication/pp1453