

Ciclo de vida de los datos: Práctica 2

Francisco Alonso Sarria

December 13, 2020

Índice

| | |
|--|-----------|
| 1 Introducción | 2 |
| 2 Librerías y funciones | 2 |
| 3 Carga de datos y exploración inicial | 4 |
| 4 Transformación e integración de variables | 5 |
| 4.1 Transformación de variables y creación de un objeto espacial | 5 |
| 4.2 Enriquecimiento del dato | 6 |
| 5 Limpieza de datos | 8 |
| 5.1 Busqueda de datos perdidos | 8 |
| 5.2 Detección de outliers | 10 |
| 6 Normalidad y homogeneidad de la varianza | 15 |
| 7 Análisis de varianza no paramétrico | 19 |
| 8 Modelo lineal global | 20 |
| 9 Modelos predictivos y validación cruzada con datos de Europa | 23 |
| 9.1 Modelo lineal | 24 |
| 9.2 Random Forest | 28 |

| | |
|---|-----------|
| 10 Imputaciones | 30 |
| 10.1 Seleccionar casos para eliminar dato | 30 |
| 10.2 Algoritmos de imputación | 30 |
| 10.3 Validación | 31 |
| 11 Conclusiones | 34 |

1 Introducción

En esta práctica trabajaremos con un data set de temperaturas mensuales en diferentes ciudades del planeta. El fichero se denomina `GlobalLandTemperaturesByCity.csv` y puede descargarse de <https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data>.

Los datos proceden de todos los países y la cobertura temporal arranca en 1750 y acaba en 2013. Las variables que incluye son ciudad, país, latitud y longitud, temperatura e incertidumbre de la temperatura. Así mismo contiene 8.599.212 filas.

El objetivo que nos planteamos es hacer un análisis exploratorio del dataset, detectar valores perdidos y *outliers*, resolver estos problemas en la medida de lo posible, analizar los patrones espacio-temporales de temperatura y finalmente obtener un modelo predictivo que permita, estimar la temperatura a partir de un conjunto de variables que representan factores astronómicos (latitud y estacionalidad) y geográficos (elevación y distancia al mar) del clima. Este modelo permitirá además determinar si existe una tendencia estadísticamente significativa al aumento de temperatura.

Puesto que el dataset descargado de Kaggle no dispone de variables relativas a los factores geográficos, tendremos que añadirlos a partir de una capa raster de elevación global de 5 minutos (1/12 grados) de resolución espacial descargada de:

```
https://www.eea.europa.eu/data-and-maps/data/world-digital-elevation-model-etopo5/
zipped-dem-geotiff-raster-geographic-tag-image-file-format-raster-data
```

El cálculo a partir de esta capa de una capa de distancias al mar se ha hecho fuera de R con GRASS, un Sistema de Información Geográfica.

2 Librerías y funciones

Vamos a trabajar con R y necesitaremos diversas librerías para trabajar con los datos. En primer lugar, trabajamos con datos espaciales, por lo que harán falta librería que permitan manejar este tipo de información: `sp` define diversos tipos de objetos con información espacial y proporciona funciones para manejárlas, `rgdal` permite a R utilizar las funciones de la librería GDAL para leer/escribir diversos formatos de datos espaciales, `raster` permite llenar, escribir y manejar capas raster de variables espaciales. Por otro lado, las fechas aparecen como cadenas de texto; la librería `lubridate`

incluye funciones que permiten convertir estas cadenas en datos de tipo fecha. Utilizaremos la librería `randomForest` para utilizar este modelo predictivo y la librería `simputation` para probar diferentes algoritmos para llenar lagunas en los datos. Finalmente, la librería `DescTools` incluye funciones para calcular la transformación Box-Cox y optimizar su parámetro λ .

```
library(rgdal)
library(raster)
library(sp)
library(lubridate)
library(DescTools)
library(randomForest)
library(simputation)
```

Por otra parte, hemos creado algunas funciones para ayudarnos al procesamiento de los datos:

- La función `readLL` lee valores de latitud o longitud codificados en la tabla como cadenas de caracteres y obtiene una codificación en grados decimales en formato numérico.

```
readLL = Vectorize(function(x) {
  n = nchar(x)
  dir = substr(x, n, n)
  x = as.numeric(substr(x, 1, n-1))
  if (dir %in% c("S", "W")) m = -1 else m = 1
  return(m*x)
})
```

- La siguiente función implementa la inversa de la transformación Box-Cox.

```
invBoxCox <- function(x, lambda) {
  if (lambda == 0) exp(x) else (lambda*x + 1)^^(1/lambda)
}
```

- La siguiente función normaliza un vector de datos y devuelve una lista con el valor mínimo y máximo de la serie original y la serie normalizada:

```
normaliza <- function(x) {
  mn = min(x, na.rm=TRUE); mx = max(x, na.rm=TRUE)
  return(list(min=mn, max=mx, nx=(x-mn) / (mx-mn) ))
```

- Finalmente, esta función deshace la transformación anterior, para ello necesita, además de la serie transformada, los valores mínimo y máximo que se utilizaron para transformarla:

```
denormaliza <- function(nx, mn, mx) return(nx * (mx - mn) + mn)
```

3 Carga de datos y exploración inicial

Leemos la tabla descargada de Kaggle

```
datos = read.table("GlobalLandTemperaturesByCity.csv",
                    header=TRUE, sep=",", quote="\")
```

Hacemos un análisis rápido para ver el número y nombres de las variables y sus tipos, el número de filas y los valores que aparecen:

```
str(datos)

## 'data.frame': 8599212 obs. of  7 variables:
## $ dt                  : chr "1743-11-01" "1743-12-01" "1744-01-01" "1744-02-01" ...
## $ AverageTemperature   : num 6.07 NA NA NA NA ...
## $ AverageTemperatureUncertainty: num 1.74 NA NA NA NA ...
## $ City                 : chr "Århus" "Århus" "Århus" "Århus" ...
## $ Country              : chr "Denmark" "Denmark" "Denmark" "Denmark" ...
## $ Latitude              : chr "57.05N" "57.05N" "57.05N" "57.05N" ...
## $ Longitude             : chr "10.33E" "10.33E" "10.33E" "10.33E" ...

summary(datos)

##      dt          AverageTemperature AverageTemperatureUncertainty 
##  Length:8599212    Min.   :-42.7      Min.   : 0.0      
##  Class :character  1st Qu.: 10.3      1st Qu.: 0.3      
##  Mode   :character Median : 18.8      Median : 0.6      
##                  Mean  : 16.7      Mean  : 1.0      
##                  3rd Qu.: 25.2      3rd Qu.: 1.3      
##                  Max.  : 39.7      Max.  :15.4      
##                  NA's  :364130     NA's  :364130    
##      City          Country          Latitude          Longitude        
##  Length:8599212    Length:8599212    Length:8599212    Length:8599212  
##  Class :character  Class :character  Class :character  Class :character 
##  Mode   :character Mode  :character  Mode  :character  Mode  :character 
## 
## 
## 
## 

dim(datos)

## [1] 8599212      7
```

Son 8599212 filas y 7 columnas. Tanto las fechas como las coordenadas aparecen como cadenas de caracteres. No parece haber valores excesivamente altos de temperatura, quizás parezcan incluso demasiado bajos, pero no hay que olvidar que se trata de medias mensuales. El mínimo de temperatura si puede ser excesivamente bajo.

4 Transformación e integración de variables

4.1 Transformación de variables y creación de un objeto espacial

Transformamos las coordenadas en datos numéricos con la función `readLL` y las fechas en año y mes con las funciones de la librería `lubridate`. Además creo las variables `absLat` como el valor absoluto de la latitud, ya que el efecto de la latitud de la temperatura podemos en principio asumirlo como simétrico, y `solst` que convierte los meses en una variable continua con valores de 1 en la estación cálida y de -1 en la estación fría. Esta variable tiene en cuenta la diferencia entre ambos hemisferios.

```
datos$Longitude = readLL(datos$Longitude)
datos$Latitude = readLL(datos$Latitude)
datos$fecha     = ymd(datos$dt)
datos$year      = year(datos$fecha)
datos$mes       = month(datos$fecha)
datos$AbsLat    = abs(datos$Latitude)
datos$solst     = ifelse(datos$Latitude<0,
                         cos(2*pi*datos$mes/12),
                         -cos(2*pi*datos$mes/12))
```

Ahora tenemos que crear un objeto espacial a partir del `data.frame`. Como es una capa de puntos, se crea un `SpatialPointsDataFrame`. Para ello basta con especificar que columnas contienen las coordenadas y especificar el sistema de referencia espacial que utilizan mediante el correspondiente código del EPSG¹. Aunque no se especifica, y es muy posible que haya diferencias de unos puntos a otros) asumiremos que todas las coordenadas se refieren al elipsoide WGS84². A escala planetaria las diferencias entre unos y otros son insignificantes.

```
coordinates(datos) =~ Longitude+Latitude
proj4string(datos)  = CRS ("+init=epsg:4326")
```

Generamos ahora un mapa con los puntos:

¹https://en.wikipedia.org/wiki/EPSC_Geodetic_Parameter_Dataset

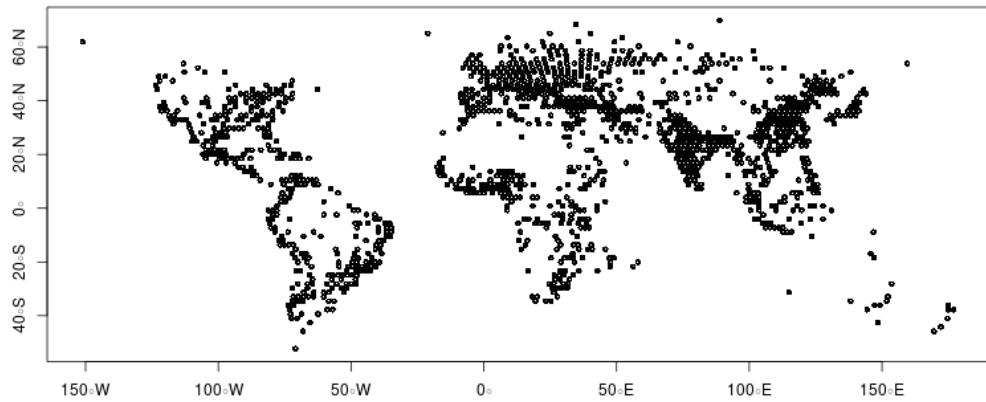
²https://en.wikipedia.org/wiki/World_Geodetic_System

```

plot(datos, axes=TRUE, cex=0.5, pch=1)

## X11cairo
## 2

```



4.2 Enriquecimiento del dato

Los datos que contiene el dataset que hemos descargado son insuficientes para hacer un modelo predictivo de la temperatura. Aunque incluye variables ligadas a factores astronómicos (estación y latitud) no incluye variables ligadas a factores geográficos (elevación o continentalidad). Por ello vamos a enriquecer el dataset con esta información a partir de una capa raster de elevación global con 5 minutos de resolución descargada de:

```

https://www.eea.europa.eu/data-and-maps/data/world-digital-elevation-model-etopo5/
zipped-dem-geotiff-raster-geographic-tag-image-file-format-raster-data

```

En primer lugar utilizamos un Sistema de Información Geográfica (GRASS) para obtener a partir de esta capa de elevaciones una capa de distancia al mar. Para ello, basta con crear una capa intermedia que contenga unos en la superficie marina y valores nulos en tierra, posteriormente se calcula, para cada celdilla, su distancia al objeto definido por las celdillas de valor 1. Las órdenes de GRASS utilizadas son:

```

r.in.gdal alwdgg_4326.tif output=alwdgg_4326
g.region rast=alwdgg_4326
r.mapcalc expression="sea;if(alwdgg_4326<0,1,null())"
r.grow.distance sea distance=distSea metric=geodesic
r.out.gdal distSea output=distSea.tif

```

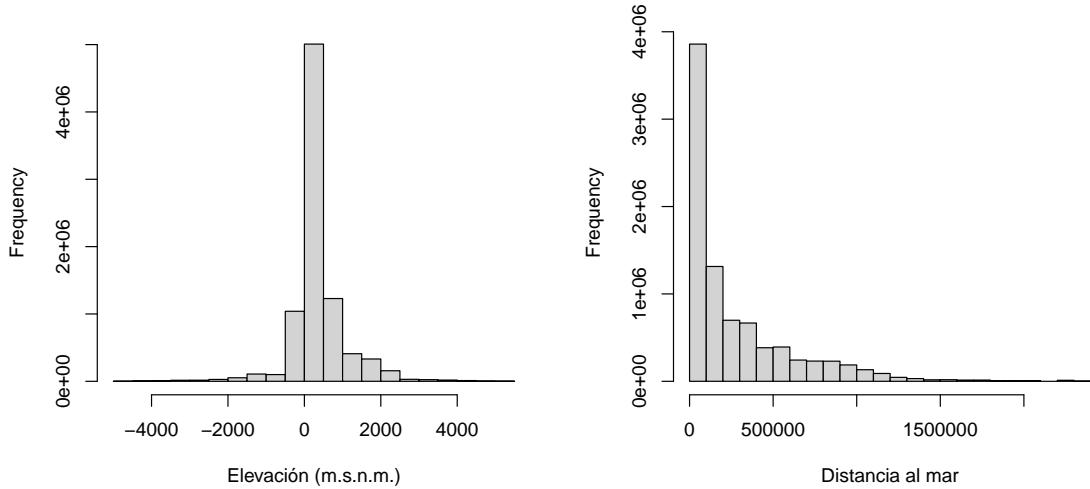
De este modo tenemos una capa de elevación y otra de distancia al mar (que puede actuar como proxy de la continentalidad). Ahora las cargamos en R, les asignamos el mismo sistema de referencia espacial que a la de los puntos:

```
MDE=raster::raster("alwdgg_4326.tif")
distSea=raster::raster("distSea.tif")
proj4string(MDE) = CRS("+init=epsg:4326")
proj4string(distSea) = CRS("+init=epsg:4326")
```

A continuación extraemos de cada capa raster los valores de elevación y distancia al mar correspondientes a los puntos y

```
datos$elev = raster::extract(MDE, datos)
datos$dist = raster::extract(distSea, datos)
```

```
par(mfrow=c(1, 2))
hist(datos$elev, main="", xlab="Elevación (m.s.n.m.)")
hist(datos$dist, main="", xlab="Distancia al mar")
```



Vemos que hay valores de elevación negativos ya que la capa de elevaciones incluye batimetría y la incertidumbre posicional de ambos dataset hace que las ciudades costeras puedan aparecer en el mar. Para corregir este error asignaremos una elevación cero a todas aquellas ciudades que aparezcan con elevación inferior a cero:

```
datos$elev = pmax(datos$elev, 0)
```

Por otro lado vemos que tanto los valores de distancia al mar como los de elevación (tras la transformación) tienen una distribución muy asimétrica. Esto es esperable ya que los datos proceden de ciudades y estas suelen concentrarse a baja altitud y cerca del mar.

5 Limpieza de datos

5.1 Busqueda de datos perdidos

En primer lugar buscaremos datos perdidos para cada una de las variables que utilizaremos posteriormente:

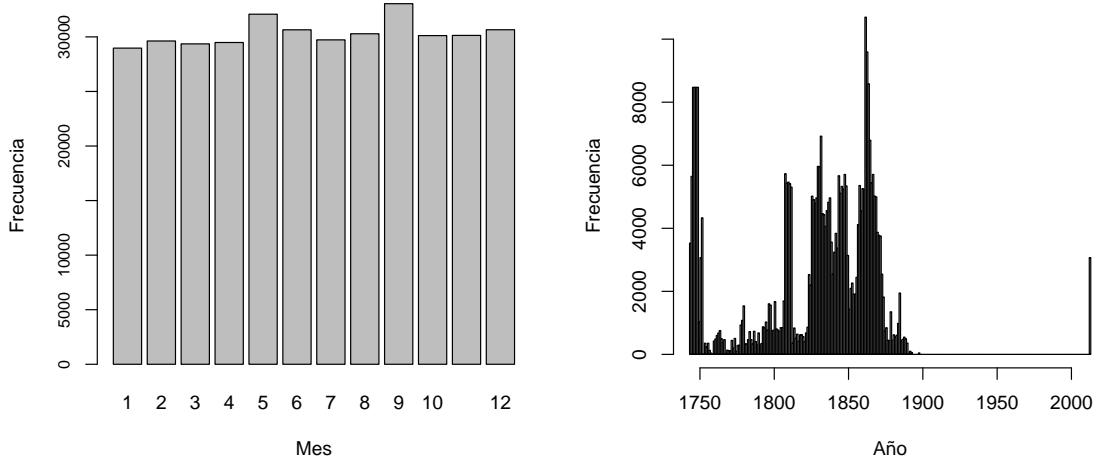
```
length(which(is.na(datos$AverageTemperature)))  
  
## [1] 364130  
  
length(which(is.na(datos$year)))  
  
## [1] 0  
  
length(which(is.na(datos$solst)))  
  
## [1] 0  
  
length(which(is.na(datos$AbsLat)))  
  
## [1] 0  
  
length(which(is.na(datos$elev)))  
  
## [1] 0  
  
length(which(is.na(datos$dist)))  
  
## [1] 0  
  
length(which(is.na(datos$year)))  
  
## [1] 0
```

Vemos que solo hay valores perdidos de temperatura,

```
wNA = which(is.na(datos$AverageTemperature))  
100*length(wNA)/nrow(datos)  
  
## [1] 4.234458
```

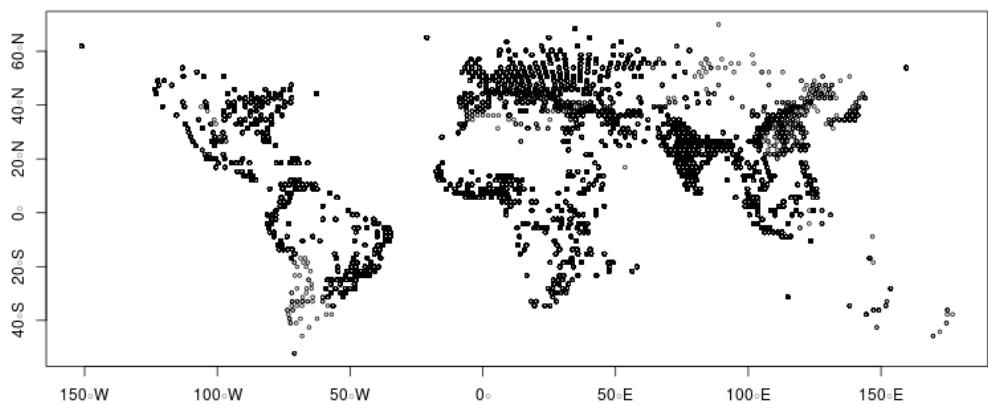
en concreto un 4.23 % de datos perdidos. Vamos por tanto a tratar de localizar cuando y donde se producen:

```
par(mfrow=c(1,2))  
barplot(table(datos@data[wNA, "mes"]), xlab="Mes", ylab="Frecuencia",  
        cex.axis=0.8)  
hist(datos@data[wNA, "year"], breaks=250,  
      main="", xlab="Año", ylab="Frecuencia")
```



Los datos perdidos se reparten homogéneamente entre los diferentes meses, pero se concentran antes de 1900 y en el año 2013. Vamos a continuación a analizar su reparto espacial:

```
plot(datos[wNA, ], axes=TRUE, cex=0.5, pch=1)
```



Parece que se distribuyen homogéneamente por todo el mundo.

A tenor de esto resultados, la solución adoptada ha sido eliminar los años anteriores a 1900 y el 2013 y concentrarnos en analizar la evolución de la temperatura en el período 1900-2012:

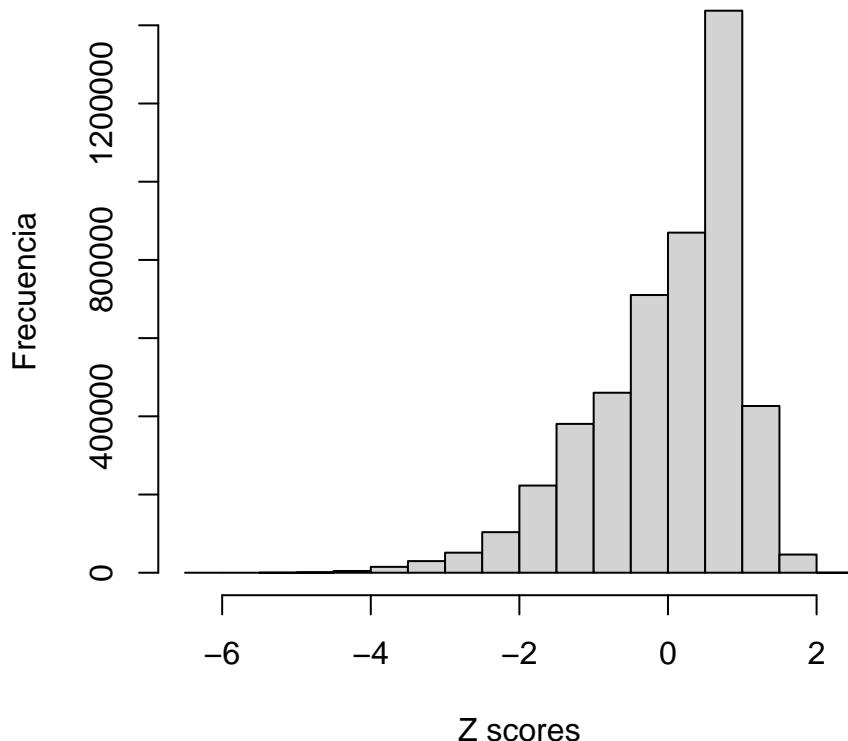
```
datos2 = datos[which(datos$year>=1900 & datos$year<2013), ]
dim(datos2)

## [1] 4759560      14
```

5.2 Detección de outliers

En primer lugar vamos a ver un histograma de los z-scores de la temperatura:

```
x = datos2$AverageTemperature
hist((x-mean(x))/sd(x), xlab="Z scores", ylab="Frecuencia", main="")
```



En principio podemos considerar que todo valor con un z-score superior a 3 en valor absoluto puede ser considerado un *outlier*. En este caso no aparecerían en el extremo derecho de la distribución pero si en el extremo izquierdo (z-scores negativos). De todos modos hay que tener en cuenta que la fuerte dependencia de las temperaturas de los meses más fríos de la elevación y de la distancia al mar y la distribución exponencial de estos factores climáticos, que hemos visto anteriormente, explicarían esta distribución.

Para tener más elementos de juicio, vamos a analizar la presencia de outliers por zona geográfica. Para ello comenzamos definiendo 5 zonas geográficas divididas por los trópicos (latitudes -23.44 y + 23.44) y los círculos polares (latitudes -66.5 y 66.5):

```
datos2$Zona = cut(datos2$Lat, c(-90, -66.5, -23.44, 23.44, 66.5, 90))
table(datos2$Zona)

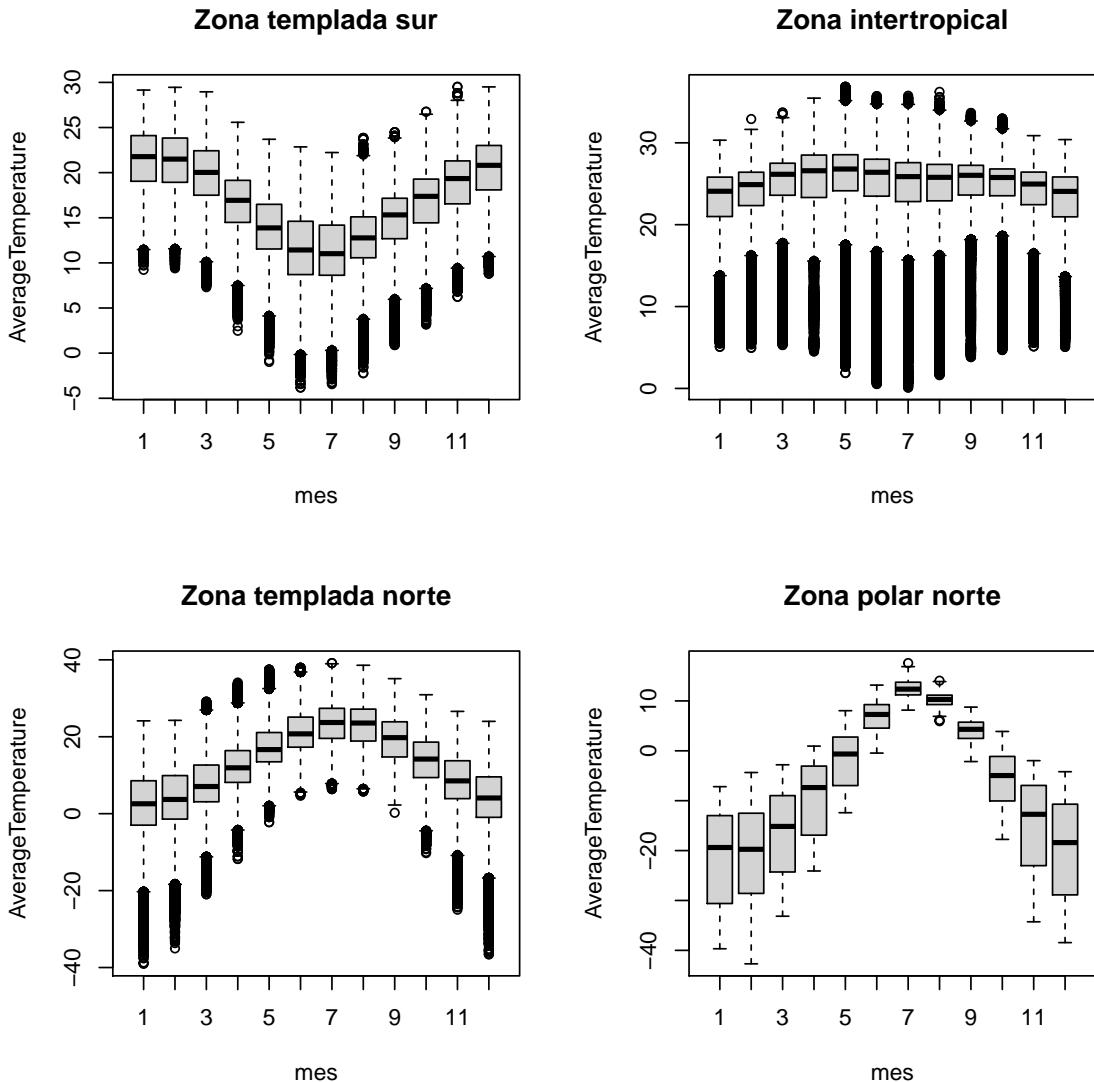
## 
## (-90, -66.5]  (-66.5, -23.4]  (-23.4, 23.4]  (23.4, 66.5]  (66.5, 90]
```

| | | | | | |
|-----|---|--------|---------|---------|------|
| # # | 0 | 227808 | 1857720 | 2671320 | 2712 |
|-----|---|--------|---------|---------|------|

Comprobamos que no hay ciudades más allá del círculo polar antártico, pero si del ártico. A continuación vamos a ver la distribución de valores por zona y por mes:

Vamos a ver outliers por mes y por zona geográfica

```
par(mfrow=c(2,2))
boxplot(AverageTemperature~mes, datos2[which(datos2$Zona=="(-66.5,-23.4]")), ,
        main="Zona templada sur")
boxplot(AverageTemperature~mes, datos2[which(datos2$Zona=="(-23.4,23.4]")), ,
        main="Zona intertropical")
boxplot(AverageTemperature~mes, datos2[which(datos2$Zona=="(23.4,66.5]")), ,
        main="Zona templada norte")
boxplot(AverageTemperature~mes, datos2[which(datos2$Zona=="(66.5,90]")), ,
        main="Zona polar norte")
```



Aparecen grupos importantes de valores extremos inferiores en 3 de las zonas, pero se trata de grupos homogéneos, sin presencia de valores aislados que puedan identificarse como valores anómalos. El hecho de que no aparezcan en la zona polar norte (donde el efecto de la distancia al mar y la elevación es mínimo) parece reforzar nuestra hipótesis de que los extremos inferiores se deben a estos factores.

Vamos a analizar ahora *outliers* por década y por zona geográfica, para ello creamos primero una nueva variable denominada década. En este caso resulta interesante añadir una línea punteada a cada gráfico marcando la media de cada una de las zonas:

```

datos2$decada = as.numeric(cut(datos2$year, seq(1899,2020,10)))

par(mfrow=c(2,2))

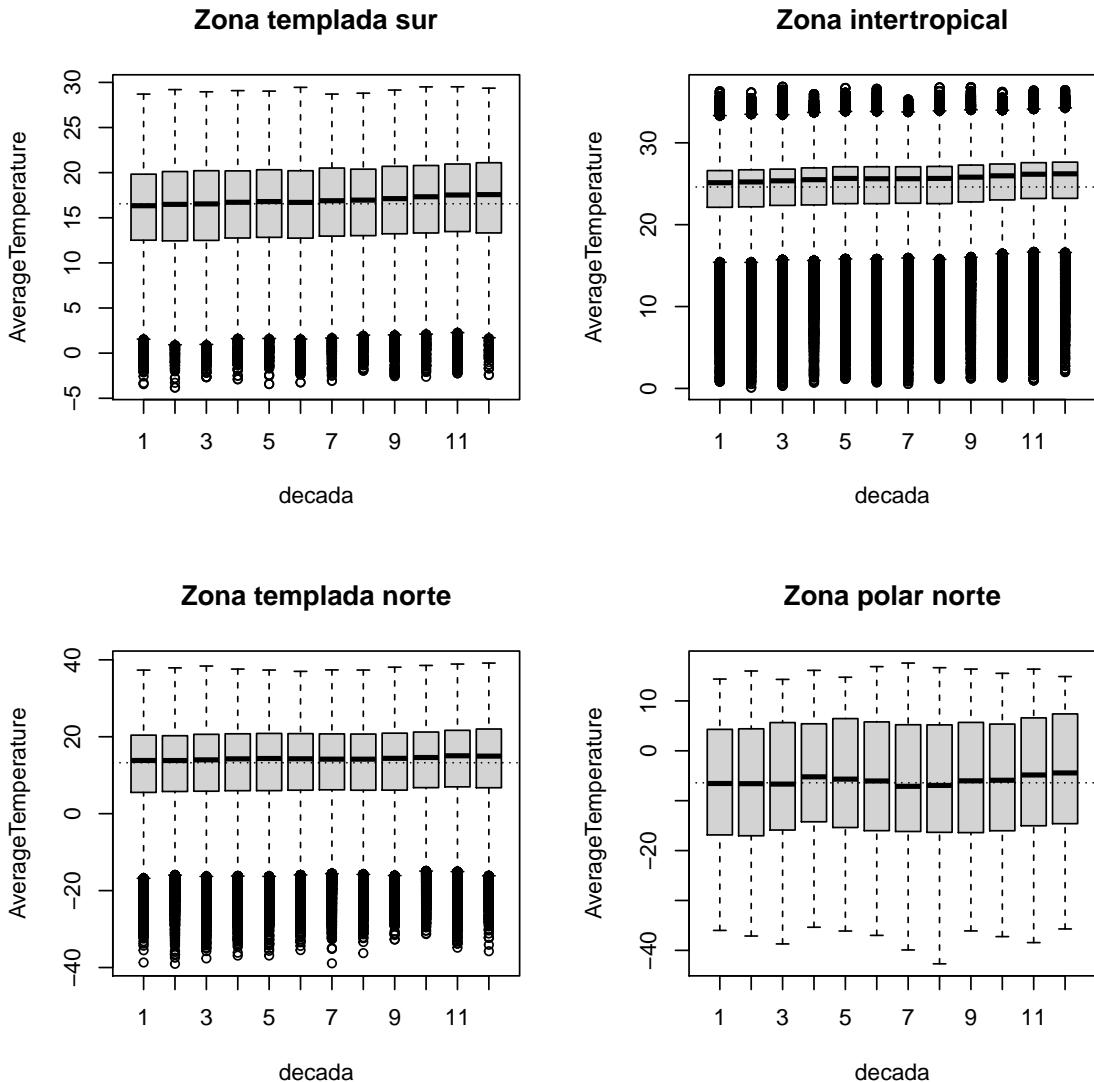
boxplot(AverageTemperature~decada,
        datos2[which(datos2$Zona=="(-66.5,-23.4]"),],
        main="Zona templada sur")
abline(h=mean(datos2@data[which(datos2$Zona=="(-66.5,-23.4]"),2]), lty=3)

boxplot(AverageTemperature~decada,
        datos2[which(datos2$Zona=="(-23.4,23.4]"),],
        main="Zona intertropical")
abline(h=mean(datos2@data[which(datos2$Zona=="(-23.4,23.4]"),2]), lty=3)

boxplot(AverageTemperature~decada,
        datos2[which(datos2$Zona=="(23.4,66.5]"),],
        main="Zona templada norte")
abline(h=mean(datos2@data[which(datos2$Zona=="(23.4,66.5]"),2]), lty=3)

boxplot(AverageTemperature~decada,
        datos2[which(datos2$Zona=="(66.5,90]"),],
        main="Zona polar norte")
abline(h=mean(datos2@data[which(datos2$Zona=="(66.5,90]"),2]), lty=3)

```



6 Normalidad y homogeneidad de la varianza

Ya hemos visto que la distribución de temperaturas está muy lejos de ser normal. A pesar de ello vamos a utilizar el test de Shapiro-Wilk para evaluar la normalidad:

```
shapiro.test(datos2$AverageTemperature[sample(1:nrow(datos2), 5000)])  
##
```

```

## Shapiro-Wilk normality test
##
## data: datos2$AverageTemperature[sample(1:nrow(datos2), 5000)]
## W = 0.92117, p-value < 2.2e-16

```

Tal como se esperaba, el p-valor es muy bajo y se debe descartar la hipótesis nula de que estos datos provengan de una distribución normal. Por tanto vamos a tratar de utilizar la transformación Box-Cox optimizando el parámetro λ :

```

lambda=BoxCoxLambda(datos2$AverageTemperature)

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length
= nonseasonal.length): NA/Inf replaced by maximum positive value

lambda

## [1] 1.402156

datos2$AT.bc = BoxCox(datos2$AverageTemperature, lambda=lambda)

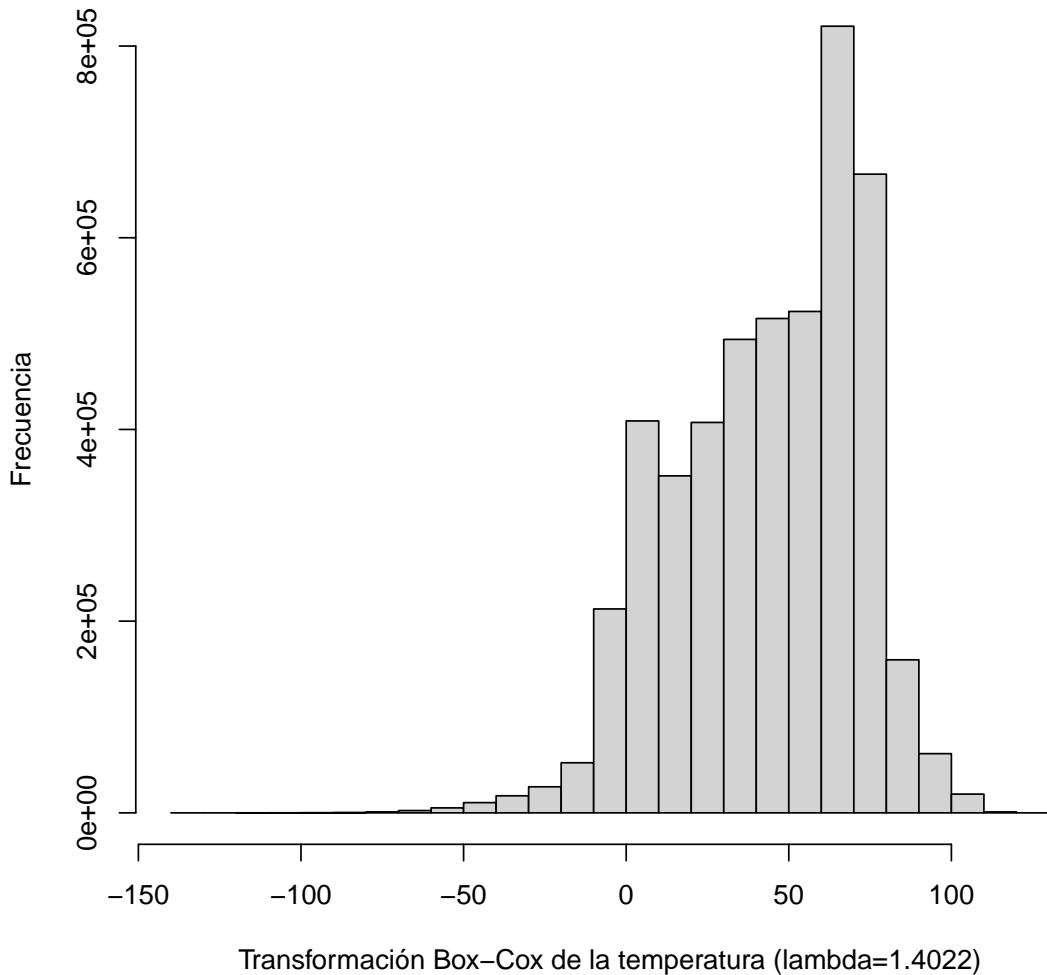
```

Vamos ahora a ver el histograma de la variable resultante y el resultado de aplicar sobre ella el test de Shapiro-Wilk:

```

14 = round(lambda, 4)
x12 = paste0("Transformación Box-Cox de la temperatura (lambda=", 14, ")")
hist(datos2$AT.bc, main="", xlab=x12, ylab="Frecuencia")

```



```
shapiro.test(datos2$AT.bc[sample(1:nrow(datos2), 5000) ] )

##
## Shapiro-Wilk normality test
##
## data: datos2$AT.bc[sample(1:nrow(datos2), 5000) ]
## W = 0.96679, p-value < 2.2e-16
```

Aunque la distribución es algo menos asimétrica, sigue estando lejos de ser normal. Los gráficos de

caja que hemos visto antes muestran también diferencias en la varianza entre los distintos meses y zonas, vamos de todos modos a evaluarlo con el test de Levene tanto para los valores como para los transformados:

```
car::leveneTest (AverageTemperature~as.factor(mes), datos2)

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group      11 103503 < 2.2e-16 ***
##             4759548
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

car::leveneTest (AverageTemperature~as.factor(Zona), datos2)

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group      3 419096 < 2.2e-16 ***
##             4759556
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

car::leveneTest (AverageTemperature~as.factor(decada), datos2)

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group      11 33.458 < 2.2e-16 ***
##             4759548
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

car::leveneTest (AT.bc~as.factor(mes), datos2)

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group      11 61300 < 2.2e-16 ***
##             4759548
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

car::leveneTest (AT.bc~as.factor(Zona), datos2)
```

```

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group      3 299195 < 2.2e-16 ***
##             4759556
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

car::leveneTest(AT.bc~as.factor(decada), datos2)

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group     11 30.067 < 2.2e-16 ***
##             4759548
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

En todos los casos se observan diferencias significativas en las varianza.

7 Análisis de varianza no paramétrico

Debido a que los datos no cumplen las condiciones para utilizar un análisis de la varianza paramétrico, utilizaremos el test de Kruskal-Wallis:

```

kruskal.test(AverageTemperature~as.factor(Zona), datos2)

##
##  Kruskal-Wallis rank sum test
##
## data: AverageTemperature by as.factor(Zona)
## Kruskal-Wallis chi-squared = 1636619, df = 3, p-value < 2.2e-16

kruskal.test(AverageTemperature~as.factor(mes), datos2)

##
##  Kruskal-Wallis rank sum test
##
## data: AverageTemperature by as.factor(mes)
## Kruskal-Wallis chi-squared = 745033, df = 11, p-value < 2.2e-16

kruskal.test(AverageTemperature~decada, datos2)

```

```

## 
## Kruskal-Wallis rank sum test
## 
## data: AverageTemperature by decada
## Kruskal-Wallis chi-squared = 8061.2, df = 11, p-value < 2.2e-16

```

que resulta ser significativo en todos los casos. A continuación utilizaremos el test de Wilcoxon para comparar las medias en décadas consecutivas y comprobar si los incrementos son constantes en el tiempo o no:

```

for (p in 1:11) {
  decadas = levels(as.factor(datos2$decada)) [c(p,p+1)]
  subset = datos2[which(datos2$decada %in% decadas),]
  wt = wilcox.test(AverageTemperature~decada, subset)
  cat(p, "-", p+1, wt$p.value, "\n")
}

## 1 - 2 0.005331193
## 2 - 3 8.891158e-25
## 3 - 4 5.898936e-15
## 4 - 5 9.910155e-12
## 5 - 6 0.3830265
## 6 - 7 0.8632119
## 7 - 8 0.7721065
## 8 - 9 3.312816e-22
## 9 - 10 2.70437e-37
## 10 - 11 3.101853e-57
## 11 - 12 0.0008805667

```

Los resultados muestran diferencias significativas en las primeras décadas (desde 1900-09 a 1940-49) y en las últimas (desde 1980-89 hasta 2000-12) mientras que en las décadas intermedias no parece haber diferencias significativas.

8 Modelo lineal global

A pesar de que los datos no son apropiados para utilizar una regresión lineal, vamos a intentarlo y a comparar los resultados con los que obtendremos mediante validación cruzada que, al ser un método más empírico de evaluación de la exactitud, no sufre los condicionamientos de los estadísticos que se calculan directamente a partir de los resultados de la regresión lineal:

```

mod1 = lm(AverageTemperature~year+solst+AbsLat+Long+elev+dist, datos2)
summary(mod1)

## 
## Call:
## lm(formula = AverageTemperature ~ year + solst + AbsLat + Long +
##     elev + dist, data = datos2)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -39.008  -3.505    0.644    4.139   17.885 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.305e+01 1.601e-01   81.53 <2e-16 ***
## year        9.933e-03 8.179e-05   121.45 <2e-16 ***
## solst       6.008e+00 3.773e-03   1592.49 <2e-16 ***
## AbsLat      -4.565e-01 1.820e-04  -2507.73 <2e-16 ***
## Long        -2.561e-03 3.663e-05   -69.90 <2e-16 ***
## elev        -3.059e-03 4.653e-06   -657.39 <2e-16 ***
## dist        -1.312e-07 8.277e-09   -15.85 <2e-16 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 5.82 on 4759553 degrees of freedom
## Multiple R-squared:  0.6635, Adjusted R-squared:  0.6635 
## F-statistic: 1.564e+06 on 6 and 4759553 DF,  p-value: < 2.2e-16

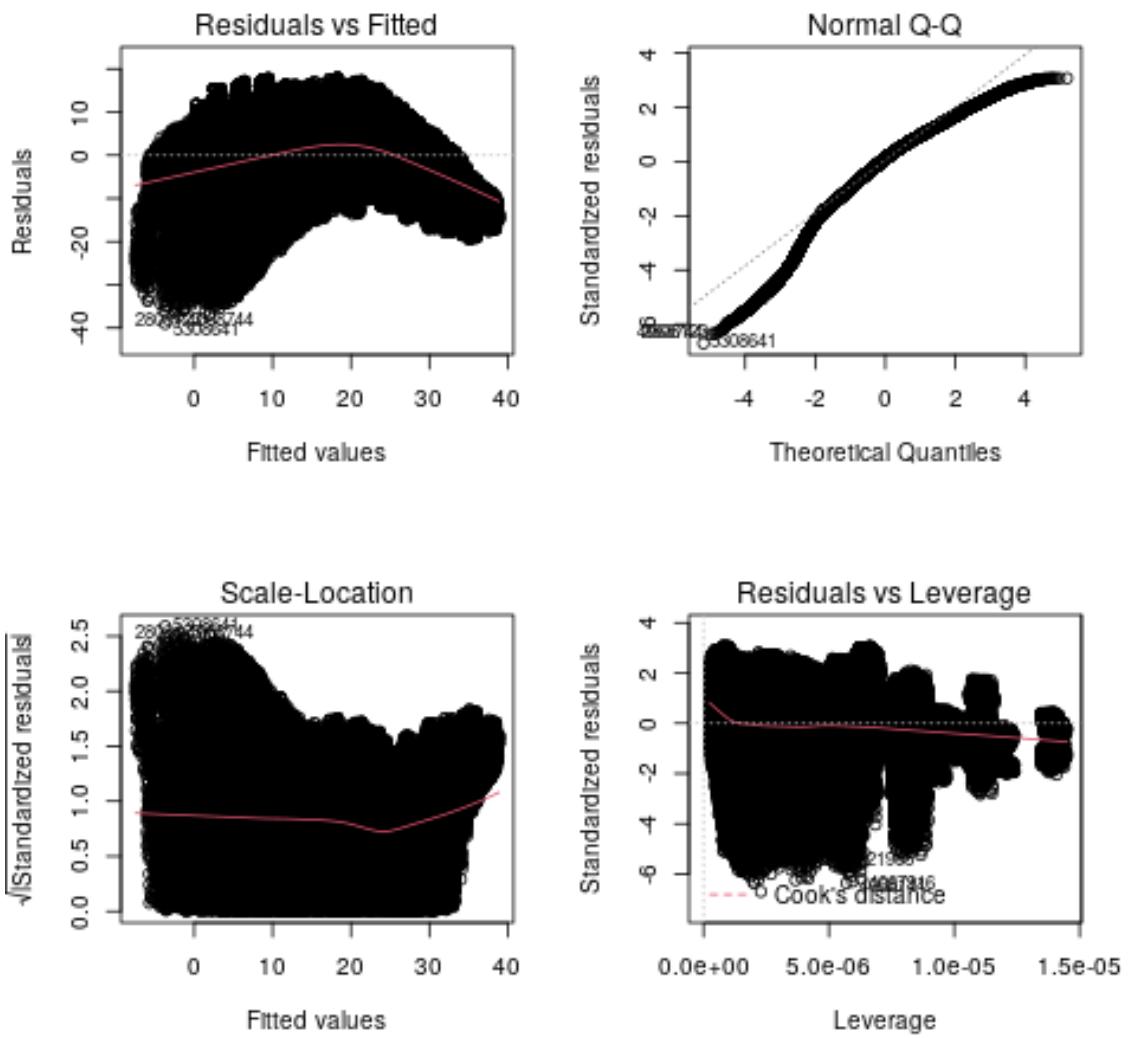
```

Todas las variables son altamente significativas y los coeficientes son los esperables. El error estándar es de 5.82 grados, que dado el rango de valores de temperatura es aceptablemente bajo, y $r^2 = 0.6635$ lo que significa que el modelo explicaría un 66.35 % de la variabilidad de la temperatura. Vamos ahora a ver los gráficos de diagnóstico de la regresión:

```

par(mfrow=c(2,2))
plot(mod1)

```



El gráfico superior izquierdo muestra los residuos contra los valores predichos, la forma de la nube de puntos revela que la relación es realmente no lineal, con lo que se viola otro de los supuestos de la regresión lineal. El gráfico superior derecho es un QQ-plot que muestra la falta de normalidad de los residuos, previsible viendo la falta de normalidad de la variable original. El gráfico inferior derecho muestra la ausencia de valores influyentes en la regresión (no aparece ninguna isolínea de distancias de Cook) lo que indica que los posibles outliers encontrados y que pensamos que son valores legítimos, no influyen en la regresión.

Estos resultados indican que los estadísticos obtenidos anteriormente no son excesivamente fiables. Vamos ahora a obtener algunos de esos estadísticos mediante validación cruzada haciendo 10 folds-CV. Es decir, el dataset se divide en 10 subgrupos y cada uno de ellos es predicho mediante un

modelo calibrado con todos los demás:

```
kf = sample(1:nrow(datos2), nrow(datos2))
cortes = round(seq(1, nrow(datos2), length=11))
pred = rep(NA, nrow(datos2))
for (f in 1:10) {
  wval = kf[cortes[f]:cortes[f+1]]
  mod = lm(AverageTemperature~year+solst+AbsLat+Long+elev+dist,
            datos2[-wval,])
  pred[wval] = predict(mod, datos2[wval,])
}
```

Con este procedimiento hemos ido rellenando un vector de valores estimados que utilizaremos ahora para estimar la exactitud utilizando r^2 , la raíz cuadrada del error cuadrático medio (RMSE) y el sesgo (media de los errores):

```
cor(pred, datos2$AverageTemperature)^2
## [1] 0.6634599
sqrt(mean((pred-datos2$AverageTemperature)^2))
## [1] 5.820089
mean(pred-datos2$AverageTemperature)
## [1] 1.205497e-06
```

Los resultados son sorprendentemente similares para r^2 y RMSE, sobre todo teniendo en cuenta que las estimaciones obtenidas directamente del modelo de regresión no son fiables y que en validación lo normal es obtener estimaciones más pesimistas de la exactitud que en calibración.

9 Modelos predictivos y validación cruzada con datos de Europa

A continuación nos centraremos en datos de Europa para comparar un modelo predictivo hecho con regresión lineal con otro hecho con Random Forest utilizando validación cruzada. La razón de limitarnos a Europa es que Random Forest es un modelo mucho más exigente en recursos informáticos y resultó imposible utilizarlo con todos los datos.

9.1 Modelo lineal

En primer lugar creamos un nuevo dataset que contenga solo los datos de Europa:

```
Europa = c("Spain", "Portugal", "France", "Italy", "Germany", "United Kingdom",
          "Ukraine", "Sweden", "Switzerland", "Serbia", "Romania", "Poland",
          "Norway", "Netherlands", "Montenegro", "Moldova", "Macedonia",
          "Latvia", "Ireland", "Hungary", "Iceland", "Greece", "Georgia",
          "Estonia", "Czech Republic", "Denmark", "Cyprus", "Croatia",
          "Bulgaria", "Bosnia And Herzegovina", "Belarus", "Belgium",
          "Austria", "Albania")
datosE = datos2[which(datos2$Country %in% Europa), ]
```

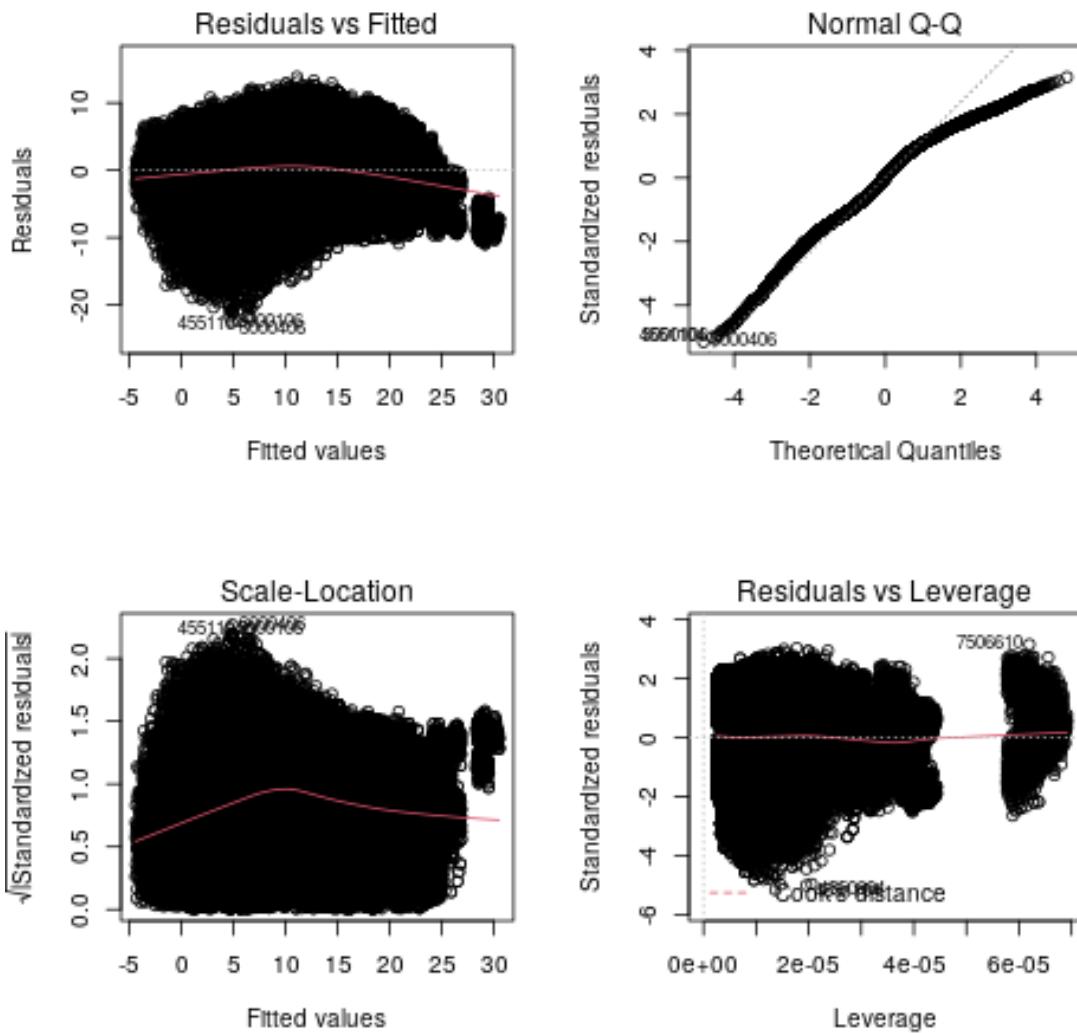
A continuación obtenemos el modelo lineal y obtenemos los estadísticos y gráficos de diagnóstico:

```
mod1E = lm(AverageTemperature~year+solst+AbsLat+Long+elev+dist, datosE)
summary(mod1E)

##
## Call:
## lm(formula = AverageTemperature ~ year + solst + AbsLat + Long +
##     elev + dist, data = datosE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -22.5747  -3.3919   0.1174   3.5937  13.8623 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.531e+01  3.259e-01   46.99 <2e-16 ***
## year        1.052e-02  1.644e-04   64.01 <2e-16 ***
## solst       7.245e+00  7.584e-03  955.28 <2e-16 ***
## AbsLat      -4.958e-01  1.071e-03 -462.78 <2e-16 ***
## Long        -4.664e-02  5.353e-04  -87.14 <2e-16 ***
## elev        -2.930e-03  1.789e-05 -163.79 <2e-16 ***
## dist        -3.920e-06  4.132e-08  -94.87 <2e-16 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.384 on 668501 degrees of freedom
## Multiple R-squared:  0.6404, Adjusted R-squared:  0.6404 
## F-statistic: 1.984e+05 on 6 and 668501 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2, 2))  
plot(mod1E)
```

Los resultados son muy similares a los obtenidos globalmente. Pasamos a hacer validación cruzada



```

kf = sample(1:nrow(datosE), nrow(datosE))
cortes = round(seq(1,nrow(datosE), length=11))
pred = rep(NA, nrow(datosE))
for (f in 1:10) {
  wval = kf[cortes[f]:cortes[f+1]]

```

```

mod = lm(AverageTemperature~year+solst+AbsLat+Long+elev+dist,
          datosE[-wval,])
pred[wval] = predict(mod, datosE[wval,])
}

```

y a evaluar los tres estadísticos de exactitud:

```

cor(pred, datosE$AverageTemperature)^2

## [1] 0.6403797

sqrt(mean((pred-datosE$AverageTemperature)^2))

## [1] 4.384497

mean(pred-datosE$AverageTemperature)

## [1] -8.651961e-06

```

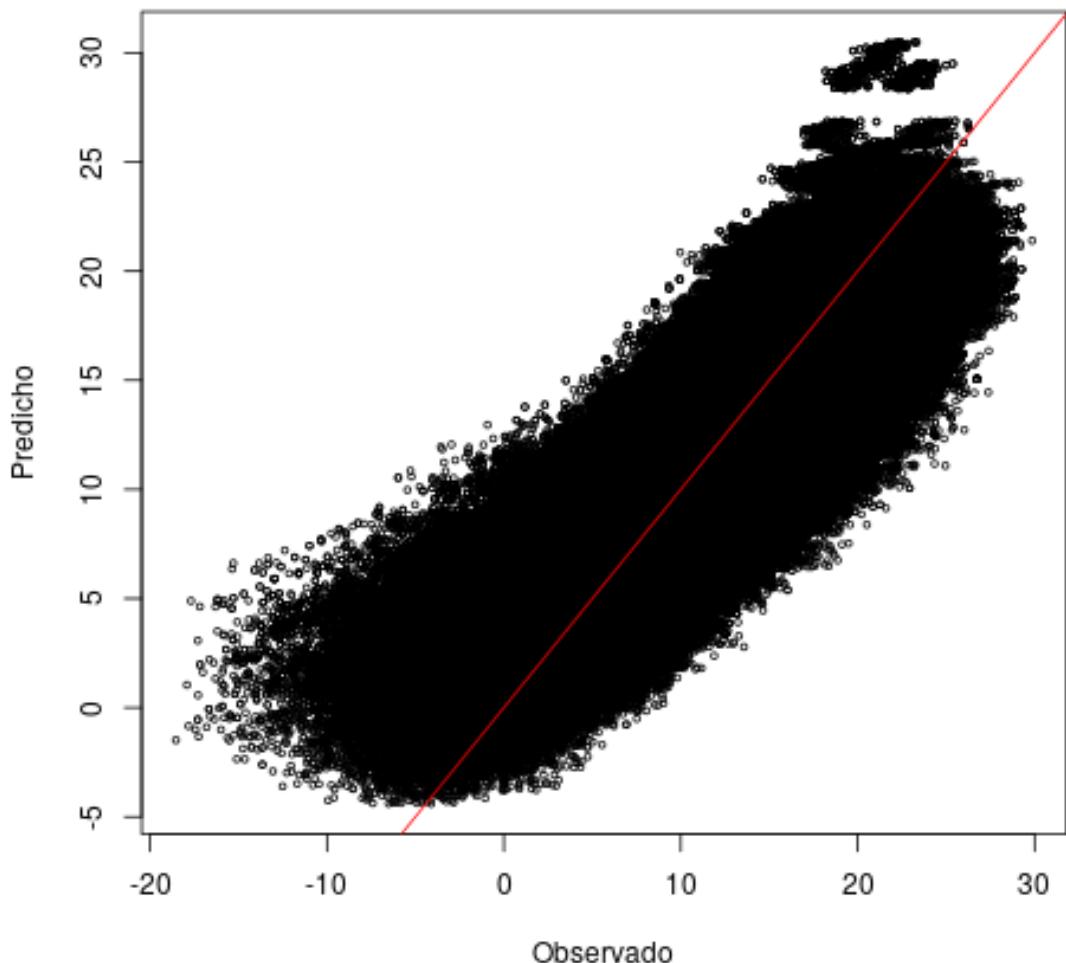
De nuevo apenas hay diferencia con los valores obtenidos en calibración ni con los resultados obtenidos con todos los datos. Vamos ahora a representar los valores observados contra los valores predichos:

```

plot(datosE$AverageTemperature, pred, cex=0.5,
      xlab="Observado", ylab="Predicho")
abline(0,1,col="red")

## X11cairo
##           2

```



Se observa una cierta no linealidad en el gráfico que refleja la que hemos visto anteriormente al analizar los gráficos de diagnóstico. Por otro lado se comprueba como la regresión lineal en realidad predice la media de Y para cada grupo de valores de X_i lo que conlleva una cierta dificultad para predecir los máximos y los mínimos de la variable. En este caso se ve especialmente bien con los mínimos ya que los valores observados llegan a -20 y los esperados no llegan a -5. La causa es que son estos valores mínimos los principales responsables de la no normalidad de la variable y por tanto aquellos para los que habrá más diferencia entre valores observados y esperados.

9.2 Random Forest

Random Forest es la

```
predRF = rep(NA, nrow(datosE))
for (f in 1:10) {
  cat(f,10, "\n")
  wval = kf[cortes[f]:cortes[f+1]]
  mod = randomForest(AverageTemperature~year+solst+AbsLat+Long+elev+dist,
                      datosE[-wval,])
  predRF[wval] = predict(mod, datosE[wval,])
  rm(mod); gc()
}
```

Vamos ahora a ver los resultados de los estadísticos de exactitud:

```
cor(predRF, datosE$AverageTemperature)^2
## [1] 0.8667122

sqrt(mean((predRF - datosE$AverageTemperature)^2))
## [1] 2.675371

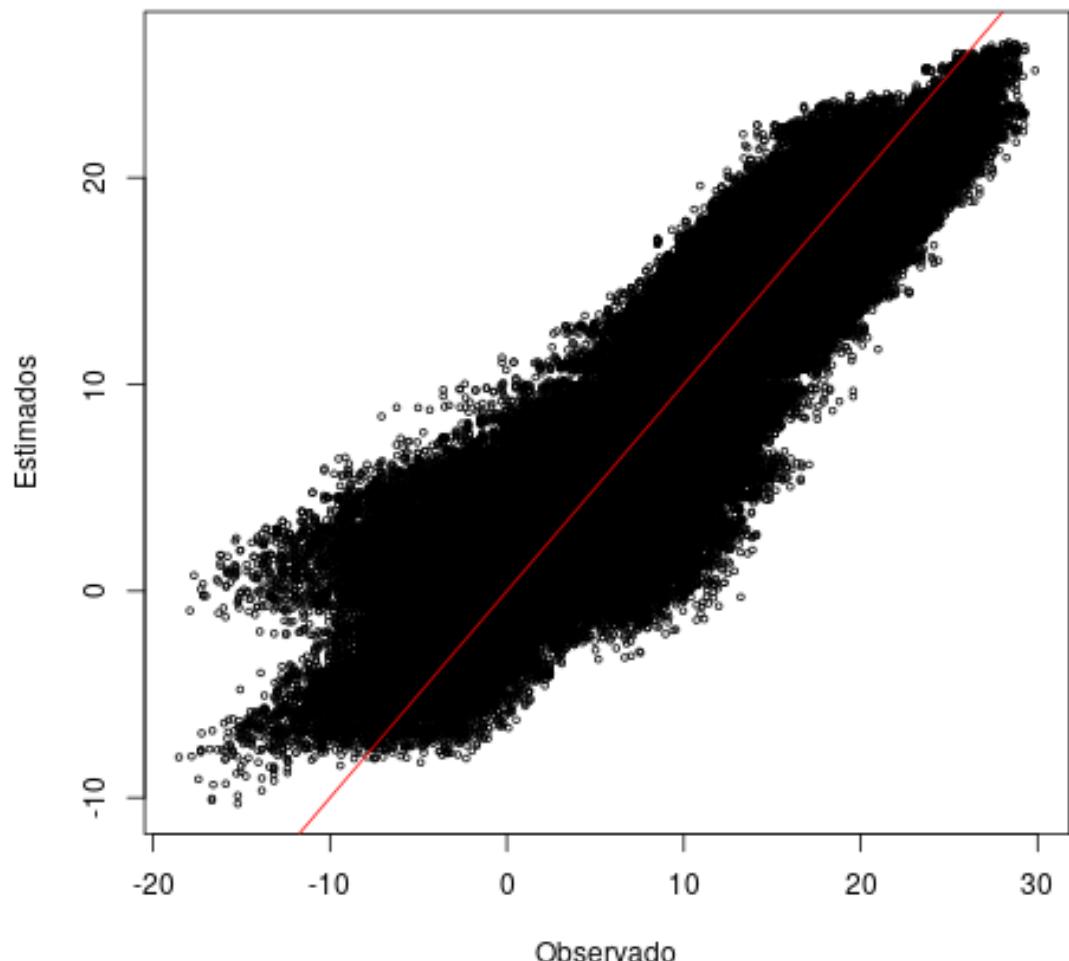
mean(predRF - datosE$AverageTemperature)
## [1] -0.001209849
```

El bias es algo mayor que en regresión lineal, pero sigue estando muy por debajo de la precisión con la que se mide la variable. Por otro lado. r^2 y RMSE están muy por debajo del caso anterior. Podemos afirmar que este modelo explica el 86.67 % de la variabilidad de la temperatura y el error medio cometido es de 2.67 grados en valor absoluto.

A continuación veremos el gráfico de valores estimados contra observados:

```
plot(datosE$AverageTemperature, pred, cex=0.5,
      xlab="Observado", ylab="Estimados")
abline(0, 1, col="red")

## X11cairo
## 2
```



Aunque random forest también comprime algo el rango de variabilidad, el efecto no es tan importante como en el caso de la regresión lineal. En este caso se alcanzan predicciones de hasta -10 °C. La relación entre valores observados y estimados es lineal, aunque con algunos "salientes" en la nube de puntos. Esto se debe a que Random Forest divide el espacio de variables en regiones con forma de "hiperparalelepípedo" que pueden dar lugar al mismo sesgo para todos los casos situados en la misma región y que acaban provocando saltos en la variable.

10 Imputaciones

Aunque la solución adoptada para resolver el problema de los datos faltantes ha sido su eliminación debido a su concentración temporal, hemos querido probar diferentes técnicas de imputación. Para ellos simularemos un 5% de datos faltantes de temperatura y los rellenaremos utilizando 7 métodos diferentes. Finalmente, puesto que tenemos los valores reales, podremos determinar el error cometido por cada método.

10.1 Seleccionar casos para eliminar dato

Creamos un nuevo dataset con los datos de Europa pero seleccionando solo las variables utilizadas en los modelos. Seleccionamos además un 5 % de los datos como datos perdidos. Guardaremos el vector de las filas con datos perdidos para un análisis posterior.

```
datosE2 = datosE[,c(2,8:11,13:15)]
miss = sample(1:nrow(datosE), round(nrow(datosE)/20))
datosE2[miss, "AverageTemperature"] = NA
```

10.2 Algoritmos de imputación

Los 7 métodos que utilizaremos son:

- Media global

```
predMissMed = rep(mean(datosE2$AverageTemperature, na.rm=TRUE),
length(miss))
```

- Mediana global

```
predMissMd = rep(median(datosE2@data$AverageTemperature, na.rm=TRUE),
length(miss))
```

- Mediana por mes y latitud

```
predMissMdCnd = impute_median(datosE2@data,
AverageTemperature ~ mes + Lat)[miss,1]
```

- Modelo lineal similar al utilizado anteriormente:

```
predMissLM = impute_lm(datosE2@data,
AverageTemperature~year+elev+solst+AbsLat+dist)[miss,1]
```

- Modelo Random Forest similar al utilizado anteriormente:

```

predMissRF = impute_rf(datosE2@data,
                         AverageTemperature~year+elev+solst+AbsLat+dist) [miss, 1]

```

- Media de los 5 vecinos más próximos. La documentación de la librería `simputation` no dice si las variables se normalizan internamente por la función `impute_knn`, así que es necesario empezar por normalizarlas:

```

normYear = normaliza(datosE2$year)
datosE2@data$year = normYear$nx

normSolst = normaliza(datosE2$solst)
datosE2$solst = normSolst$nx

normAbsLat = normaliza(datosE2$AbsLat)
datosE2$AbsLat = normAbsLat$nx

normElev = normaliza(datosE2$elev)
datosE2$elev = normElev$nx

normDist = normaliza(datosE2$dist)
datosE2$dist = normDist$nx

```

```

predMiss5NN = impute_knn(datosE2@data,
                           AverageTemperature~year+elev+solst+AbsLat+dist, k=5) [miss, 1]

```

- Media de los 20 vecinos más próximos.

```

predMiss20NN = impute_knn(datosE2@data,
                           AverageTemperature~year+elev+solst+AbsLat+dist, k=20) [miss, 1]

```

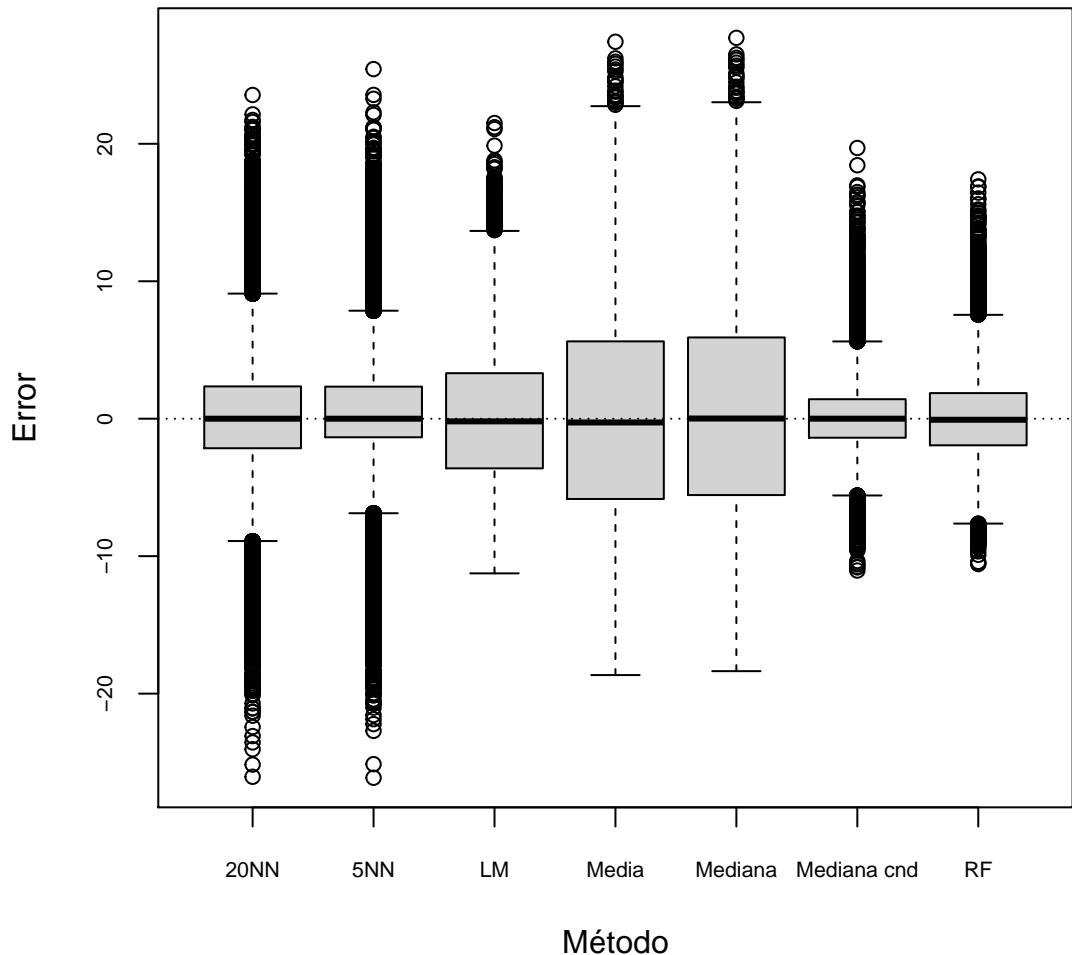
10.3 Validación

Para terminar calcularemos el error cometido en cada uno de los casos perdidos por cada uno de los métodos. Los resultados se presentan con un gráfico de cajas y los valores de los tres estadísticos de exactitud utilizados: r^2 , RMSE y bias:

```

errores = c(predMissMed - datosE$AverageTemperature[miss],
            predMissMd - datosE$AverageTemperature[miss],
            predMissMdCnd - datosE$AverageTemperature[miss],
            predMissLM - datosE$AverageTemperature[miss],
            predMissRF - datosE$AverageTemperature[miss],
            predMiss5NN - datosE$AverageTemperature[miss],
            predMiss20NN - datosE$AverageTemperature[miss])
metodos = rep(c("Media", "Mediana", "Mediana cnd", "LM", "RF", "5NN", "20NN"),
                each=length(miss))
boxplot(errores~metodos, ylab="Error", xlab="Método", cex.axis=0.7)
abline(h=0, lty=3)

```



Viendo en gráfico parece que el mejor método es el de mediana condicionada. Pasamos a calcular los estadísticos de exactitud:

```
acc = matrix(rep(NA, 21), ncol=7)
rownames(acc) = c("r2", "RMSE", "bias")
colnames(acc) = c("Media", "Mediana", "Mediana cnd", "LM", "RF", "5NN", "20NN")

acc[1, 1] = cor(predMissMed, datosE$AverageTemperature[miss])^2

## Warning in cor(predMissMed, datosE$AverageTemperature[miss]): the
```

```

standard deviation is zero

acc[1,2] = cor(predMissMd, datosE$AverageTemperature[miss])^2

## Warning in cor(predMissMd, datosE$AverageTemperature[miss]): the
## standard deviation is zero

acc[1,3] = cor(predMissMdCnd, datosE$AverageTemperature[miss])^2
acc[1,4] = cor(predMissLM, datosE$AverageTemperature[miss])^2
acc[1,5] = cor(predMissRF, datosE$AverageTemperature[miss])^2
acc[1,6] = cor(predMiss5NN, datosE$AverageTemperature[miss])^2
acc[1,7] = cor(predMiss20NN, datosE$AverageTemperature[miss])^2

acc[2,1] = sqrt(mean((predMissMed - datosE$AverageTemperature[miss])^2))
acc[2,2] = sqrt(mean((predMissMd - datosE$AverageTemperature[miss])^2))
acc[2,3] = sqrt(mean((predMissMdCnd - datosE$AverageTemperature[miss])^2))
acc[2,4] = sqrt(mean((predMissLM - datosE$AverageTemperature[miss])^2))
acc[2,5] = sqrt(mean((predMissRF - datosE$AverageTemperature[miss])^2))
acc[2,6] = sqrt(mean((predMiss5NN - datosE$AverageTemperature[miss])^2))
acc[2,7] = sqrt(mean((predMiss20NN - datosE$AverageTemperature[miss])^2))

acc[3,1] = mean(predMissMed - datosE$AverageTemperature[miss])
acc[3,2] = mean(predMissMd - datosE$AverageTemperature[miss])
acc[3,3] = mean(predMissMdCnd - datosE$AverageTemperature[miss])
acc[3,4] = mean(predMissLM - datosE$AverageTemperature[miss])
acc[3,5] = mean(predMissRF - datosE$AverageTemperature[miss])
acc[3,6] = mean(predMiss5NN - datosE$AverageTemperature[miss])
acc[3,7] = mean(predMiss20NN - datosE$AverageTemperature[miss])

options(width=220)
acc

##           Media   Mediana Mediana cnd          LM          RF          5NN        20NN
## r2            NA         NA  0.8810444  0.641023559  0.84024461  0.4729543  0.49921013
## RMSE    7.34435917 7.3494658  2.5380374  4.400367419  3.02951339  5.8110557  5.63400176
## bias  -0.01136948 0.2741629  0.1200911 -0.005502866 -0.00805994  0.2087925  0.08139378

```

No puede obtenerse r^2 para los métodos de media y mediana globales ya que al ser todas las predicciones iguales, su desviación típica es 0.

Los peores resultados se obtienen, como era de esperar, con los métodos de media y mediana globales. El método de regresión global obtiene mejores resultados que los de vecinos más próximos

(siendo más rápido de ejecutar). El método de mediana condicionada tiene los mejores r^2 y RMSE pero su sesgo es relativamente alto, algo mayor que la precisión con la que se mide la variable. Random Forest tiene el sesgo más bajo de todos los métodos y es el segundo en r^2 y RMSE pero a costa de ser el más lento. Por tanto el método recomendable en este caso es el de mediana condicionada.

11 Conclusiones