

Chirps-validation.R

Check point
1 : df-pol-chirps.RData

§1. Carga los datos

- Pop_col pcp-col.xlsx
- Chirps_df chirps-ts.parquet

§2. Carga datos elevación de datos abiertos

- SRTM_30 SRTM_30.col1.tif

§3. Convierte a objeto sf (Simple features) df with geom col.

- Chirps_point \leftarrow st_as_sf(chirps_df,...)
- Sttns.points \leftarrow st_as_sf(pcp_col,...)

§4. Extrae la altitud de los puntos de las sttns

- SRTM_30.sttns \leftarrow extract(SRTM_30, sttns.points)

El nuevo archivo de estaciones tiene la col.
altitud, chequear dif. respecto al SRTM

Bias	Q1	Q2	Q3	Nan
PBIas	-4.1	4.3	24.6	~7.7% ± 60 y PBIas ± 25
	-9.4%	9.8%	2.6%	Sobreestimación (varias series)

§5. Función para extraer los pxls CHIRPS cercanos

- sttns.points \$r = 0.05 (en concordancia con la resolución espacial de chirps)

- pol_ \leftarrow function(x, r){...}

- pol_list = list()

for (i in 1:nrow(sttns.points)) {

pol_list[[i]] = pol_(sttns.points[i],

5.1 Agrega ID de la sttns

sttns.points\$p[1:i] =

sttns.points\$CodeEstacion[i]

Cada fila del objeto sttns.points es una estación.mes;
 la función pol_:

- Crea un polígono alrededor de la coord. \$geometry
- Convierte a un objeto sf y hace joins espaciales st_join
- Calcula distancias st_distance y ordena por distancia
- selecciona el 1er elemento y filtra el objeto Chirps.point

Ajustes

§5

51 col \$ID a c/elemento de pol_list

- [DEPRECATED]
- + Col: \$nrow= 2 pol_list
 - + Vector ID_px comparando \$nrow_ y \$n. chirps en c.e. de pol_list
 - + Cambiar la col: \$r para las estaciones de San Andrés
 - + aplicar nuevamente la función pol_ a los elementos de ID_px

Actualizar

Sttns-IDEAM.R

§1. Concatena los csv

- df_sttns /TS-climatic/ideam

87330 x 3

Obs: sttns-mes

§2. join con diccionario de sttns (CNE-IDEAM)

- CNE-IDEAM CNE-IDEAM.xlsx

4509 x 20

sttns

§3. Aplica criterios de sel de sttns (completitud > 90%)

- df_sttns2 \leftarrow pivot_wider (dplyr::df1, CNE-IDEAM))

501 012 x 20

Sttns_pcp.col-train.parquet

df de 1009 filas x 486 cols, filas son sttns filtradas por criterio de completitud con excepción de 8 de los del oriente con baja densidad de sttns x Sup km^2

71 sttns ~7%

y 480 cols meses de

1981-01 a 2020-12 y 6 cols; Col: name, dpto, n, sttn, lat, lon

Seleccionar el max con una función
no permite mantener la col \$geometry
para filtrar en el obj Chirps.point

Checkpoint 2 :

.RData

§6. join con stlns.points

```
• pol2_list = list()
  for (i in 1:length(stlns.points)) {
    # join para traer las cols de stlns (obs)
    pol2_list[[i]] = left_join(pol2_list[[i]], stlns.points)
```

§7. Pivotea cols a filas

```
for (i in 1:length(pol2_list)) {
  # Selecciona cols, pivotea cols a filas y join para
  # conformar un df con cols ID, date, CHIRPS y stlns.
  pol2_list[[i]] = left_join(
    # geometry-x: CHIRPS .dplyr::(pol2_list[[i]]),
    # geometry-y: stlns .dplyr::(pol2_list[[i]]))
  # CHIRPS hasta 2022, mientras stlns (train) hasta 2020-12
```

§8. Evalua la estacionariedad de las series

• 8.1 función para obtener Orden AR

```
get_ar_order <- function(x, diff = ...) { ... }
```

La función ajusta un modelo AR para obtener el orden, el cual se usará como arg. lag de la prueba ADF.

diff para convertir TS # Sin diff, dado que
a estacionaria y poder se parte de lo que
ajustar "correctamente" modelo AR se quiere fugar,
TS climáticas son para aplicar la prueba ADF
no-estacionarias se requiere obtener
debido a la comp estacional. el arg lag

• 8.2 función prueba `stlts::adf.test`

```
ADF-test <- function(x, m.order, type) { ... }
```

la función corre la prueba ADF, usando el arg lag
obtenido con `get_ar_order`, devuelve un df con las cols:

- + lag_t ; rezago donde p.value es máx.
- + P.value ; p.value máx
- + RD.t ; regla decisión

Ajustes

S8.

8.1 función `get_ar_order`

8.2 función `ADF-test`

Apéndice 2: Prueba ADF a ts original

TS original: Chirps-v2-0

obtener el orden del modelo AR
el orden será el parámetro de la prueba ADF

```
m-order = get_ar_order(x = pol2_list, diff = TRUE)
# Sin diff ordinaria: diff = FALSE
```

gráfico stlns con Error al ajustar modelo AR

tal vez por
 TS no-estacionaria → se podría
 diff doble

```
df-ADF = ADF-test(x = pol2_list, m.order, type = 'type1')
# Sin diff ordinalmente m.order &
```

Urce::Ur.df

Apéndice 3 : Prueba ADF a ts descomposición base STL

o 8.3 df-pgram : periodograma

```
len_ = length(Pol2.list)
df-pgram = data.frame(freq = rep(0, len_),
                      Period = rep(0, len_))
```

Calcula el periodo del ciclo (oculto) estacional a partir de la f.densidad espectral

```
for (i in 1:len_) {
```

```
    pgram = spectrum(Pol2.list[[i]]$chirps-v2.0)
    f-max = which.max(pgram$spec)
    df-pgram$freq[i] = pgram$freq[f-max]
    df-pgram$period[i] = round(1/pgram$freq[f-max])
}
```

gráfico muestra Stl-ts con periodo 4 y 6

o 8.4 d-chirps.v2.0 : lista ts diff estacionalmente

```
d-chirps.v2.0 = list()
```

lista con ts deseestacionalizada

```
for (i in 1:len_) {
  d-chirps.v2.0 = data.frame(
    chirps.col = diff(Pol2.list[[i]]$chirps-v2.0),
    lag = df-pgram$period(i)) %>%
    rename(`chirps-v2.0` = chirps.col)
}
```

Crea col con ts deseestacionalizada

```
for (i in 1:len_) {
  Pol2.list[[i]]$d-chirps.v2.0 =
    c(rep(NA, df-pgram$period(i)),
      d-chirps.v2.0[[i]]$chirps-v2.0)
}
```

o 8.5 obtiene orden AR usando ts.diff estacionalmente

```
m_order4 = get_ar_order(x = d-chirps.v2.0, diff_ = FALSE)
```

o 8.6 Prueba stl::adf.test a ts.diff estacionalmente

```
df-ADF4 = ADF-test(d-chirps.v2.0, m_order4, type = 'type1')
```

8.7 Urca :: ur.df

TS descomposición STL

```
Stl-chirps.v2.0 = list()
```

```
for (i in 1:len_) {
  y <- tsibble(Pol2.list[[i]] %>% dplyr
               functions)
  fit1 <- stl(y, S.window = "periodic")
  # remueve la comp estacional
  Stl-chirps.v2.0[[i]] = data.frame(
    `chirps-v2.0` = y$`chirps-v2.0` - fit1$time.series[, "seasonal"])
}
```

```
m_order6 = get_ar_order(x = Stl-chirps.v2.0, diff_ = FALSE)
# Con diff ordinaria: diff_ = TRUE
```

```
df-ADF6 = ADF-test(x = Stl-chirps.v2.0, m_order6, type = 'type1')
# Con diff ordinalmente m_order5
```

Urca::ur.df

Apéndice 4 : Prueba ADF a ts descomposición feast STL

TS descomposición feast_stl

```
f-Stl-chirps.v2.0 = list()
```

```
for (i in 1:len_) {
  y <- tsibble(Pol2.list[[i]] %>% dplyr
               functions)
  fit2 <- y |>
    model(STL(`chirps-v2.0` ~
              season(window = "periodic"),
              robust = TRUE)) |>
    Components()
  f-Stl-chirps.v2.0[[i]] = data.frame(
    `chirps-v2.0` = y$`chirps-v2.0` - fit2$`season.year`)
}
```

```
m_order8 = get_ar_order(x = f-Stl-chirps.v2.0, diff_ = FALSE)
# Con diff ordinaria: diff_ = TRUE
```

```
df-ADF8 = ADF-test(x = f-Stl-chirps.v2.0, m_order8, type = 'type1')
# Con diff ordinalmente m_order7
```

Urca::ur.df

No Recorre 2to : 604 veces
Concluye TS no-estacionaria

No Recorre 2to : 596 veces
Concluye TS no-estacionaria

§9. Calcula medidas de desempeño

```

pol2.list[[i]]$mes = format( $Date, "%B")
$col_mes = ifelse( $mes %in% DTF,
                   'DTF', ifelse(...))

R_pearson = cor()
R_Spearman = cor()
for (i in 1:len_) {
  R_pearson[i] = Cor( `chirps-v2-o`, $stlns,
                      use = "complete.obs")
  R_Spearman[i] = Cor( `chirps-v2-o`, $stlns,
                       use = "complete.obs",
                       method = "spearman")
}
d_stlns = list()
d_stlns[i] = data.frame(
  stlns = diff($stlns,
               lag = df_pgram$period))
# Crea col d_stlns
$d_stlns = c(rep(NA, df_pgram$period),
            d_stlns$stlns)
d_R_pearson = cor()
d_R_pearson[i] = Cor( d_chirps.v2.0[i]$chirps.v2.o,
                      d_stlns[i]$stlns)

df_pearson = data.frame(r = c(R_pearson,
                               d_R_pearson),
                        Serie = c("original",
                                  "d.f estacional"))

# grafico Boxplot Coef Corr Pearson
# grafico Boxplot Coef Corr Spearman
# test de Significancia Coef Corr Spearman

```

R_Spearman_mes

```

R_Spearman_mes = list()
meses = unique($mes)
for (i in 1:len_) {
  Corrs = sapply(meses, function(m) {
    df_mes <- subset(Pol2.list[[i]], mes == m)
    cor_value <- cor(df_mes$d.chirps.v2.o,
                      df_mes$d.stlns,
                      method = "spearman")
    return(cor_value) # vector 1x12
  })
  R_Spearman_mes[i] <- Corrs
}

# Boxplot por mes
# Boxplot por col-mes
# Coef corr Spearman Cruzado

```

Apéndice

```

CCf_spearman <- function(x, y, max_lag = 10) {
  X_rank = x
  Y_rank = y
  lags <- seq(-max_lag, max_lag)
  cor_values <- sapply(lags, function(lag) {
    if (lag < 0) {
      Cor(X_rank[1:(length(x)+lag)], Y_rank[(1-lag):length(y)],
           ..., method = "spearman")
    } else if (lag > 0) {
      Cor(X_rank[(1+lag):length(x)], Y_rank[1:(length(y)-lag)],
           ..., method = "spearman")
    } else {
      Cor(X_rank, Y_rank, ..., method = "spearman")
    }
  })
  df_cor <- as.data.frame(t(cor_values))
  colnames(df_cor) <- paste0("Lag-", lags)
  return(df_cor)
}

# qmap::fitQmap b. 755-830

```

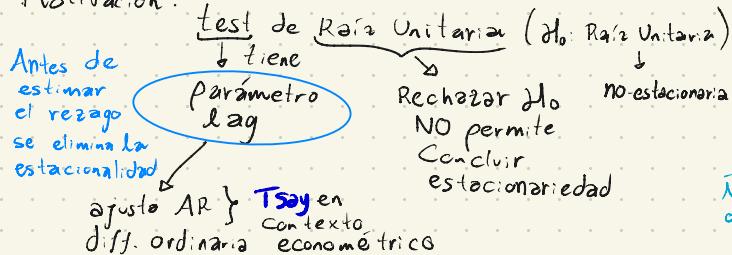
○ Chirps_Validation.R

SB. Evalua la estacionariedad de las series

Si hay una estacionalidad determinística el proceso puede ser no-estacionario
Pero la prueba ADF

Concluye estacionario

Motivación:



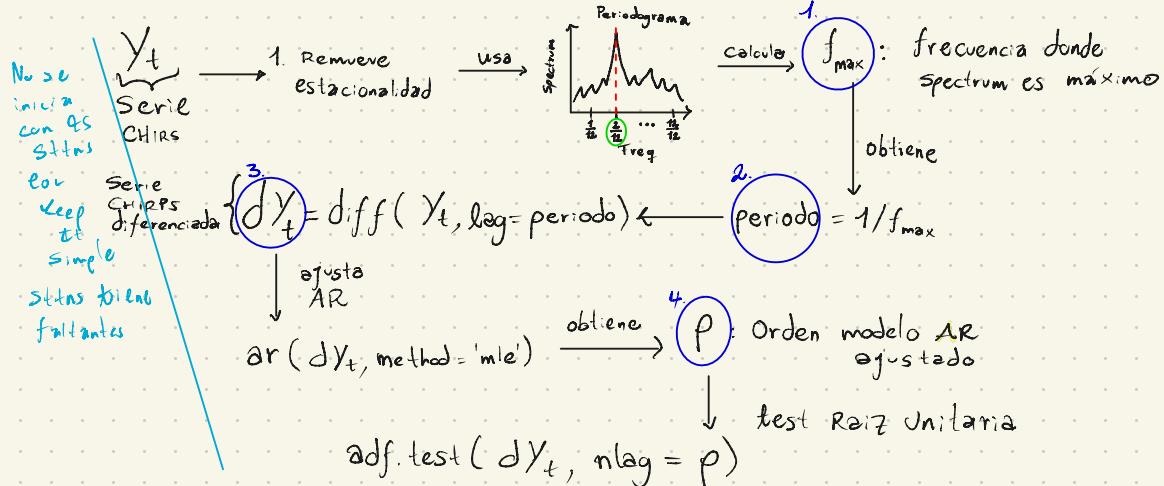
estructuras autocorrelación temporal → No cumple Supuesto independencia

↓
No cumple Supuesto i.i.d.

↓
Puede llevar a relaciones espurias

Uso correcto corr.

Initial choice of parameter «lag»



Rechaza H_0 :

Si $P\text{-value} < 0.05$

$f(dY_t, dZ_t)$ → Performance

Argumentar

- i) Si afirmar que es estacionaria y no tener Raíz Unitaria
- ii) Cuál es la ventaja?
- iii) mantiene propiedades estadísticas?
- iv) no requiere diff. ordinaria para ajustar modelo ARIMA

las métricas se aprox. más al valor real del parámetro y NO Sobreestima/Subestima

Statistical methods for climate scientist

5.5 Intro. to stochastic Process

5.2 Stochastic Processes. (Pg. 102)

most climate TS are nonstationary, because they depend on calendar day and time. Winter and summer temp have different dist.

Unit Root Test in Time Series

5.6 Dickey-Fuller and related Tests

6.3 DF tests for a Unit Root (Pg. 205)

o Chirps-validation.R

§3 diff estacional

Motivación: Al aplicar la prueba ADF a la TS original, luego de ajustar un modelo AR (Tsay, 2014) y obtener el orden de este modelo para usarlo como arg $\log d$ dado que la prueba ADF es sensible al orden, se concluye que No rechaza H_0 , luego hay raíz unitaria por tanto la TS es no-estacionaria, lo que lleva a que las métricas no se aprox. al valor real y puedan ser subestimadas/ Sobreestimadas debido a las relaciones espurias (uso correcto de la cor...)

Uno de los motivos por los que la serie es no-estacionaria es debido a la componente estacional determinística, por lo cual en el contexto de

In the area of time series called Spectral analysis

trend stationarity

¿Cuál es la relación entre la prueba de raíz Unitaria y la estacionariedad?

yo

Por qué la teoría

Spectral analysis for Stationary time series?
yo Y si es no-estacionaria?

El periodograma puede verse como una herramienta para la detección de posibles ciclos (ocultos) deterministas en una serie temporal.

Una forma de eliminar la comp. estacional es usando diferencias estacionales $(1 - B^s)^D$, donde s es el periodo del ciclo estacional.

Para encontrar s y proceder a eliminar la comp. estacional;

1. Se determina el valor de la freq. donde se maximiza el periodograma de la serie, un valor alto de periodograma para una f_j está asociado al Ciclo estacional que se quiere eliminar de la TS.

2. Calcula el periodo $s = 1/f_{\max I(f)}$

Por qué el máx del Periodograma?

yo

Fourier Analysis of Time Series

5.5 The Fast Fourier Transform

base teórica de R stats::specgram