StableClim - local regression analysis

Stuart C Brown

Contents

Local regressions

Code to calculate grid-based regressions coefficients for trend, variability, and SNR in air-temperature and precipitation for the spliced CMIP5 historical/RCP scenarios.

The code below reads in each of the 19 pre-processed spliced CMIP5 historical/RCP scenarios, and then for maximally overlapping 100-yr windows performs cell based regression using GLS (OLS if GLS fails).

Note, this process is repeated for the TraCE-21ka paleo-climate simulation.

Set-up

Set up the various directories etc. that will be used in the processing. Need a list of model names, rcp scenarios, and variables to iterate through

```
library(data.table)
library(raster)
library(ncdf4)
library(nlme)
library(gtools)

# List of models
model_list <- list.dirs("../data/CMIP5/historical/ts/",
  full.names = FALSE,
  recursive = FALSE
)
# Remove the CNRM-CM5 model
## Issue with volcanic forcing (see https://doi.org/10.1073/pnas.1210514109)
model_list <- model_list[!grepl("CNRM-CM5", model_list)]
scenarios <- c("rcp26", "rcp45", "rcp60", "rcp85")
clim_vars <- c("ts", "pr")
```

Regression functions

These functions perform the GLS(AR1) regressions on the global weighted mean temperatures. If for some reason the GLS(AR1) model fails, an equivalent model using OLS will be run instead.

```
# Pixel based regressions
## GLS regression
funTrAR1 <- function(x, na.rm = TRUE, ...) {
  require(nlme)
  if (sum(is.na(x)) >= 1) {
    NA
  } else {
    mod <- nlme::gls(x ~ Year,
      na = na.omit, method = "REML", verbose = FALSE,
      correlation = corAR1(
        form = ~Year, fixed = FALSE
      ),
      control = glsControl(
        tolerance = 1e-3, msTol = 1e-3,
        opt = c("nlminb", "optim"),
        singular.ok = TRUE, maxIter = 10000, msMaxIter = 10000
      )
    )
    slope <- mod$coefficients[2]
    names(slope) <- "Trend"
    var <- sd(mod$residuals)
    names(var) <- "Var"
    snr <- abs(slope) / var
    names(snr) <- "SNR"
    # return all the layers
    return(c(slope, var, snr))
  }
}
## OLS regression
funTrOLS <- function(x, na.rm = TRUE, ...) {
  if (sum(is.na(x)) >= 1) {
    NA
  } else {
    mod <- stats::lm(x ~ Year, na.action = na.omit)
    slope <- mod$coefficients[2]
    names(slope) <- "TrendOLS"
    var <- sd(mod$residuals)
    names(var) <- "VariabilityOLS"
    snr <- slope / var
    names(snr) <- "SNROLS"
    return(c(slope, var, snr))
  }
}
```

Peforming pixel regressions

This section of code performs global gls(AR1) regressions for each cell of the RCP data.

```
for (m in seq_along(model_list)) {
  model <- model_list[m]
  message("Processing model: ", model)
  m_files <- list.files("C:/tmp/cdo_outputs/", "\\.nc$", full.names = TRUE)
  m_files <- m_files[grepl("regridAnnAvg", m_files)]
  m_sub <- gsub(
    "\\(", "\\\\(",
    gsub("\\)", "\\\\)", model)
  )
  m_files <- m_files[grepl(m_sub, m_files)]
  for (scen in scenarios) {
    message("\tScenario: ", scen)
    s_files <- m_files[grepl(scen, m_files)]
    ## if no files for the scenario, skip to next scenario
    if (length(s_files) == 0) {
      next()
    }
    for (var in clim_vars) {
      message("\t\tVariable: ", var)
      v_files <- s_files[grepl(paste0("_", var, "_"), s_files)]
      stopifnot(length(v_files) == 1)
      if (var == "pr") {
        ## multiply by 365 to get yearly totals
        clim_ras <- readAll(brick(v_files)) * 365
        ## set z-index and names
        clim_ras <- setZ(
          clim_ras, seq(1850, by = 1, length.out = nlayers(clim_ras)),
          "Year"
        )
        names(clim_ras) <- paste0("X", getZ(clim_ras))
        ridx <- which(getZ(clim_ras) > 2100)
        if (length(ridx) != 0) {
          clim_ras <- clim_ras[[-ridx]]
          clim_ras <- setZ(
            clim_ras, seq(1850, by = 1, length.out = nlayers(clim_ras)),
            "Year"
          )
          names(clim_ras) <- paste0("X", getZ(clim_ras))
        }
      } else {
        clim_ras <- readAll(brick(v_files))
        ## set z-index and names
        clim_ras <- setZ(
          clim_ras, seq(1850, by = 1, length.out = nlayers(clim_ras)),
```

```
      "Year"
    )
    names(clim_ras) <- paste0("X", getZ(clim_ras))
    ridx <- which(getZ(clim_ras) > 2100)
    if (length(ridx) != 0) {
      clim_ras <- clim_ras[[-ridx]]
      clim_ras <- setZ(
        clim_ras, seq(1850, by = 1, length.out = nlayers(clim_ras)),
        "Year"
      )
      names(clim_ras) <- paste0("X", getZ(clim_ras))
    }
  }
  ## Change z-index for GFDL model
  if (model == "GFDL-CM3") {
    clim_ras <- setZ(
      clim_ras, seq(1860, by = 1, length.out = nlayers(clim_ras)),
      "Year"
    )
    names(clim_ras) <- paste0("X", getZ(clim_ras))
    ridx <- which(getZ(clim_ras) > 2100)
    if (length(ridx) != 0) {
      clim_ras <- clim_ras[[-ridx]]
      clim_ras <- setZ(
        clim_ras,
        seq(1860, by = 1, length.out = nlayers(clim_ras)),
        "Year"
      )
      names(clim_ras) <- paste0("X", getZ(clim_ras))
    }
  }
  if (grepl("HadGEM2", model)) {
    ## drop first year of Hadley data
    clim_ras <- clim_ras[[-1]]
    clim_ras <- setZ(
      clim_ras,
      seq(1860, by = 1, length.out = nlayers(clim_ras)),
      "Year"
    )
    names(clim_ras) <- paste0("X", getZ(clim_ras))
    ridx <- which(getZ(clim_ras) > 2100)
    if (length(ridx) != 0) {
      clim_ras <- clim_ras[[-ridx]]
      clim_ras <- setZ(
        clim_ras, seq(1860, by = 1, length.out = nlayers(clim_ras)),
        "Year"
      )
```

```
    names(clim_ras) <- paste0("X", getZ(clim_ras))
  }
}
## stop if non-finite or NA values present
stopifnot(all(is.finite(values(clim_ras))))
stopifnot(all(!is.na(values(clim_ras))))
## define the windows and starting positions
window <- 100
step <- 1
starts <- seq(1, to = nlayers(clim_ras) - (window - 1), by = step)
## Loop through the starts in parallel and perform regressions
# parallel progress bar
pb <- txtProgressBar(max = length(starts), style = 3)
progress <- function(n) setTxtProgressBar(pb, n)
opts <- list(progress = progress)
cls <- detectCores() - 1
cls <- makeCluster(cls)
registerDoSNOW(cls)
### Local regressions
foreach(
  s = starts, .inorder = TRUE, .packages = c("raster", "nlme"),
  .errorhandling = "remove", .options.snow = opts
) %dopar% {
  e <- s + window - 1
  sRas <- clim_ras[[s:e]]
  Year <- getZ(sRas)
  stopifnot(nlayers(sRas) == 100)
  stopifnot(length(Year) == 100)
  ## if output rasters exist, skip to next
  if (sum(
    file.exists(
      paste0(
        "../data/outputs/",
        paste(scen, var, model, sep = "/"), "/",
        paste(model, c("Trend", "Var", "SNR"), var, Year[1],
          Year[100],
          sep = "_"
        ), ".grd"
      )
    )
  ) == 3) {
    next()
  }
  ## cell based regressions
  cell_reg <- tryCatch(
    raster::calc(sRas, fun = funTrAR1),
    error = function(err) {
```

```
      err
    }
  )
  ## if GLS fails use OLS and log error
  if (inherits(cell_reg, "error")) {
    sink(paste0(
      "../data/processingErrors/localOLSErrors_",
      model, ".txt"
    ), append = TRUE)
    cat(paste0(
      "Used OLS regression (grid-cell) for model: ",
      model, ". Scenario: ", scen, ". Variable: ",
      var, ". Window starting: ", Year[1], ".\n"
    ))
    sink()
    cell_reg <- tryCatch(
      raster::calc(sRas, fun = funTrOLS),
      error = function(err) {
        err
      }
    )
  }
  ## if OLS fails - stop. There is something wrong with the data!
  if (inherits(cell_reg, "error")) {
    stop(paste0(
      "Regressions failed for model: ",
      model, ". Scenario: ", scen, ". Variable: ",
      var, ". Window starting: ", Year[1]
    ))
  }
  names(cell_reg) <- paste(c("Trend", "Var", "SNR"), var,
    Year[1], Year[100],
    sep = "_"
  )
  ## write raster outputs
  outRasFiles <- list(cell_reg[[1]], cell_reg[[2]], cell_reg[[3]])
  names(outRasFiles) <- paste(model, c("Trend", "Var", "SNR"),
    var, Year[1], Year[100],
    sep = "_"
  )
  mapply(writeRaster, outRasFiles,
    paste0(
      "../data/outputs/",
      paste(scen, var, model, sep = "/"), "/", names(outRasFiles)
    ),
    format = "raster", datatype = "FLT4S", overwrite = FALSE
  )
```

```
      }
      stopCluster(cls)
      rm(pb)
      gc()
      message("\nDone\n")
    }
  }
}
```

Combine all results and export to netcdf

Now that all the trends, variability, and SNR have been calculated locally for all grid-cells we need to export all the data from R-raster to netcdf as ensemble means.

Can use `calc(x = subset(x, which(getZ(x) == i)), fun = median, na.rm = TRUE)` for ensemble median if desired.

```
## Function for ensemble means
## for some reason zApply and stackApply kep failing?
clim_averages <- function(x, ...) {
  stopifnot(!is.null(getZ(x)))
  r_list <- list()
  for (i in unique(getZ(x))) {
    r_list[[length(r_list) + 1]] <- calc(
      x = subset(x, which(getZ(x) == i)),
      fun = mean, na.rm = TRUE
    )
  }
  s <- readAll(stack(r_list, quick = TRUE))
  s <- setZ(s, unique(getZ(x)), "Year")
  names(s) <- unique(getZ(x))
  return(s)
}

## List all files
raster_list <- list(
  "rcp26" =
    mixedsort(
      list.files("../data/Outputs/rcp26/",
        "\\.grd",
        full.names = TRUE,
        recursive = TRUE
      ),
      decreasing = FALSE
    ),
  "rcp45" =
    mixedsort(
      list.files("../data/Outputs/rcp45/",
        "\\.grd",
        full.names = TRUE,
        recursive = TRUE
      ),
      decreasing = FALSE
    ),
  "rcp60" =
    mixedsort(
      list.files("../data/Outputs/rcp60/",
        "\\.grd",
```

```
        full.names = TRUE,
        recursive = TRUE
      ),
      decreasing = FALSE
    ),
  "rcp85" =
    mixedsort(
      list.files("../data/Outputs/rcp85/",
        "\\.grd",
        full.names = TRUE,
        recursive = TRUE
      ),
      decreasing = FALSE
    )
)

for (scen in names(raster_list)) {
  message(scen)
  files <- raster_list[[scen]]
  ts <- files[grepl(paste0(scen, "//ts/"), files)]
  pr <- files[!grepl(paste0(scen, "//ts/"), files)]
  DT_ts <- data.table("file" = ts)
  DT_ts <-
    DT_ts[, "Year" := as.integer(unlist(lapply(lapply(
      strsplit(DT_ts$file, "_"),
      tail,
      n = 2
    ), head, n = 1)))][order(Year, decreasing = FALSE), ]
  DT_pr <- data.table("file" = pr)
  DT_pr <-
    DT_pr[, "Year" := as.integer(unlist(lapply(lapply(
      strsplit(DT_pr$file, "_"),
      tail,
      n = 2
    ), head, n = 1)))][order(Year, decreasing = FALSE), ]
  ## Trends
  ts_trend <- DT_ts[grepl("_Trend_", DT_ts$file)]
  pr_trend <- DT_pr[grepl("_Trend_", DT_pr$file)]
  ## Var
  ts_var <- DT_ts[grepl("_Var_", DT_ts$file)]
  pr_var <- DT_pr[grepl("_Var_", DT_pr$file)]
  ## SNR
  ts_snr <- DT_ts[grepl("_SNR_", DT_ts$file)]
  pr_snr <- DT_pr[grepl("_SNR_", DT_pr$file)]
  ## Stacks
  ts_trendR <- readAll(stack(ts_trend$file, quick = TRUE))
  ts_trendR <- setZ(ts_trendR, z = ts_trend$Year, "Year")
```

```r
stopifnot(all(is.finite(values(ts_trendR))))
pr_trendR <- stack(pr_trend$file, quick = TRUE)
pr_trendR <- setZ(pr_trendR, z = pr_trend$Year, "Year")
ts_varR <- stack(ts_var$file, quick = TRUE)
ts_varR <- setZ(ts_varR, z = ts_var$Year, "Year")
pr_varR <- stack(pr_var$file, quick = TRUE)
pr_varR <- setZ(pr_varR, z = pr_var$Year, "Year")
ts_snrR <- stack(ts_snr$file, quick = TRUE)
ts_snrR <- setZ(ts_snrR, z = ts_snr$Year, "Year")
pr_snrR <- stack(pr_snr$file, quick = TRUE)
pr_snrR <- setZ(pr_snrR, z = pr_snr$Year, "Year")
stopifnot(
  compareRaster(ts_trendR, pr_trendR, ts_varR, pr_varR, ts_snrR, pr_snrR)
)
## EnsMeans
ts_ensMeanTrend <- clim_averages(ts_trendR)
pr_ensMeanTrend <- clim_averages(pr_trendR)
ts_ensMeanVar <- clim_averages(ts_varR)
pr_ensMeanVar <- clim_averages(pr_varR)
ts_ensMeanSNR <- clim_averages(ts_snrR)
pr_ensMeanSNR <- clim_averages(pr_snrR)
stopifnot(sum(sapply(
  list(
    ts_ensMeanTrend, pr_ensMeanTrend,
    ts_ensMeanVar, pr_ensMeanVar,
    ts_ensMeanSNR, pr_ensMeanSNR
  ),
  function(x) all(is.finite(values(x)))
)) == 6)
## Write each raster to the netcdf
## Two files - one for ts and one for pr
ts_filename <-
  paste0("../data/Outputs/Final_DB/StableClim_Regression_", scen, "_ts.nc")
pr_filename <-
  paste0("../data/Outputs/Final_DB/StableClim_Regression_", scen, "_pr.nc")
## Longitude and Latitude data
xvals <- unique(values(init(ts_ensMeanTrend[[1]], "x")))
yvals <- unique(values(init(ts_ensMeanTrend[[1]], "y")))
nx <- length(xvals)
ny <- length(yvals)
lon <- ncdim_def(name = "longitude", units = "degrees_east", vals = xvals)
lat <- ncdim_def("latitude", "degrees_north", yvals)
## Missing value to use
mv <- -9999
## Time component
time <- ncdim_def(
  name = "Time",
```

```
  units = "year",
  vals = as.integer(getZ(ts_ensMeanTrend)),
  unlim = TRUE,
  calendar = "no_leap",
  longname = "Year"
)
## Define the trend variables
var_trend <- ncvar_def(
  name = "ts_trend",
  units = "degC/year",
  dim = list(lon, lat, time),
  longname = "(1) Trend",
  missval = mv
)
var_trendpr <- ncvar_def(
  name = "pr_trend",
  units = "mm/year",
  dim = list(lon, lat, time),
  longname = "(1) Trend",
  missval = mv
)
## Define the variability variables
var_var <- ncvar_def(
  name = "ts_variability",
  units = "unitless",
  dim = list(lon, lat, time),
  longname = "(2) Variability",
  missval = mv
)
## Define the SNR variables
var_snr <- ncvar_def(
  name = "ts_snr",
  units = "unitless",
  dim = list(lon, lat, time),
  longname = "(3) Signal:Noise",
  missval = mv
)
## Write data to file
ncout_ts <- nc_create(ts_filename,
  list(var_trend, var_var, var_snr),
  force_v4 = TRUE
)
## add global attributes
ncatt_put(
  ncout_ts, 0, "Title",
  paste0("Trend and variability for air temperature under ", scen)
)
```

```
ncatt_put(ncout_ts, 0, "Creation date", "2020-20-03")
ncatt_put(ncout_ts, 0, "Author", "Stuart C Brown")
ncatt_put(ncout_ts, 0, "Email", "s.brown@adelaide.edu.au")
ncatt_put(ncout_ts, 0, "Notes", "Trends (1) are reported as °C/Year,
          but represent the slope of a regression line through each 100-year
          overlapping window.\nVariability (2) is the S.D of the residuals
          from the model used to calculate the trend(1)\nSNR is the
          Signal:Noise ratio, and is absolute trend(1) divided
          by variability(3)")
## Write data to file
ncout_pr <- nc_create(pr_filename,
  list(var_trendpr, var_var, var_snr),
  force_v4 = TRUE
)
## add global attributes
ncatt_put(ncout_pr, 0, "Title", paste0("Trend and variability for
                                       precipitation under ", scen))
ncatt_put(ncout_pr, 0, "Creation date", "2020-20-02")
ncatt_put(ncout_pr, 0, "Author", "Stuart C Brown")
ncatt_put(ncout_pr, 0, "Email", "s.brown@adelaide.edu.au")
ncatt_put(ncout_pr, 0, "Notes", "Trends (1) are reported as mm/Year, but
          represent the slope of a regression line through each 100-year
          overlapping window.\nVariability (2) is the S.D of the residuals
          from the model used to calculate the trend(1)\nSNR is the
          Signal:Noise ratio, and is absolute trend(1) divided
          by variability(3)")
# Place the precip and tmax values in the file
pb <- txtProgressBar(style = 3, min = 0, max = nlayers(ts_ensMeanTrend))
for (i in 1:nlayers(ts_ensMeanTrend)) {
  ## write ts
  ncvar_put(
    nc = ncout_ts,
    varid = var_trend,
    vals = values(ts_ensMeanTrend[[i]]),
    start = c(1, 1, i),
    count = c(-1, -1, 1)
  )
  ncvar_put(
    nc = ncout_ts,
    varid = var_var,
    vals = values(ts_ensMeanVar[[i]]),
    start = c(1, 1, i),
    count = c(-1, -1, 1)
  )
  ncvar_put(
    nc = ncout_ts,
    varid = var_snr,
```

```
      vals = values(ts_ensMeanSNR[[i]]),
      start = c(1, 1, i),
      count = c(-1, -1, 1)
    )
    ## write pr
    ncvar_put(
      nc = ncout_pr,
      varid = var_trendpr,
      vals = values(pr_ensMeanTrend[[i]]),
      start = c(1, 1, i),
      count = c(-1, -1, 1)
    )
    ncvar_put(
      nc = ncout_pr,
      varid = var_var,
      vals = values(pr_ensMeanVar[[i]]),
      start = c(1, 1, i),
      count = c(-1, -1, 1)
    )
    ncvar_put(
      nc = ncout_pr,
      varid = var_snr,
      vals = values(pr_ensMeanSNR[[i]]),
      start = c(1, 1, i),
      count = c(-1, -1, 1)
    )
    ## progress
    setTxtProgressBar(pb, i)
  }
  close(pb)
  ## close the netcdf file when finished
  nc_close(ncout_ts)
  nc_close(ncout_pr)
}
```