Synchronized failure of global crop production
Mehrabi and Ramankutty 2019

zia.mehrabi@ubc.ca

March 27, 2019

Contents

1 Aim of document

The aim of this document is to provide the Supplementary Information for Mehrabi and Ramankutty 2018. This Supplementary Document was created with `knitr` [12], a software that combines the typesetting system LaTeX and the statistical computing language **R** [6], and allows for full reproduction of our results.

This document is split into three sections. In section 2 we illustrate how we make our analysis fully reproducible. Section 3 explains a bit of background to the data we are working with and shows how we manipulated the data to get it ready for analysis. Finally, section 4 walks step by step through the analysis that underpins the results presented in our paper.

2 Checkpoint

Here we call the **R** package `checkpoint` [4]. This will create a local library on your computer and install a copy of the packages required by this project as they existed on CRAN on the specified snapshot date, and update the **R** session to use these packages. This helps make our analysis fully reproducible on your machine.

```
require(checkpoint)
checkpoint(snapshotDate = "2019-02-1")
```

Note that the **R** version used here is 3.5.1 (2018-07-02). Using other versions of **R** should not have any influence on the results obtained. The function call just above installed all packages used in this document, as available on Febuary 1st 2019 in the "home" directory of my computer.

3 Data preparation

3.1 Production data

The production data we will be using for this analysis is Deepak Ray and colleagues data, which was described in detail in [8] and [7]. Briefly, these are globally representative, census data on the area, yield, of four major commodity crops (rice, maize, wheat, soybean), for the years 1961-2008. Deepak Ray and others [7] gridded this census data across the earths surface to 0.5 degree tiles, and we will be using this gridded product for the analysis in this paper.

Deepak Ray and colleagues data is not in the correct format for the analysis we wish to perform. First the data is in netcdf (.nc) format, and housed in separate years, and we need to import this into **R** and analyse it as a seamless time series. Second, the data is presented as separate Area and Yield files for each crop, and we will need to compute Production. Third, we wish to scale the data from its current spatial scale of c. $10km^2$ at the equator to (i.e. 0.083 degrees) to c. $100km^2$ (i.e. 1 degree). And finally, as we want to work with tiles of comparable spatial extent, we need to reproject the data to an equal area grid. Each of these steps are outlined below. Most of this data preparation is not evaluated in this document, although this can be easily switched by changing the 'evaluate=FALSE' argument in the `knitr` headers of the code sections to 'evaluate=TRUE'.

3.1.1 Get data function

Below is a function to read in the netcdf data from multiple files and create a common data frame containing all of the data in them. Deepak Ray's data has the same lat long referencing and time periods for all datasets, which makes this step of manipulation quite easy.

```
getdata <- function(path = "", pat = "") {
    setwd(path)
    file.list = list.files(pattern = pat)
    files = lapply(file.list, nc_open)  #get all the files
    All = NULL
    for (i in 1:length(files)) {
        Data <- as.vector(ncvar_get(files[[i]], "Data"))  #pull in data variable
        All <- cbind(All, Data)  #bind data variables together

    }
    colnames(All) <- file.list  #assign data nc file source info
    lapply(files, nc_close)  #close nc files
    All
}
```

3.1.2 Create production variable

Below is the code to create a production estimate based on Fractional area (percent under cultivation) and yield (tonnes/ha). Some crops have multiple cropping seasons, which is why their F. Area is greater than 1. We have set crops to a maximum of two cropping seasons per year (e.g. F. area of 2), except rice which is set to three. Fractional area's less than 0.01 percent are removed. We took these two steps to ensure only sensible area values included. Finally production files for each crop are saved to their own `.rds` file.

```
library(ncdf4)  #load libaries
library(raster)
library(plyr)

# set out the paths for crops.
crops <- list("soybean", "maize", "rice", "wheat")
folder <- "/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/Deepak's historical data/"  #path to raw data
newfolder <- "/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/Deepak's historical data/processed/"  #new folder for output

for (i in 1:length(crops)) {
    path = paste(folder, crops[i], sep = "")
    Area <- getdata(path = path, pat = "*Area_ver8.nc")  #this is in tonnes/ha
    Yield <- getdata(path = path, pat = "*Yield_ver8.nc")  #in % area under cultivation in grid square

    # make sure area <0.01 and >2 are not present in the data, these values make
    # little sense
    Area[Area < 0.01] <- NA
    ifelse(path == "/Users/ziamehrabi/Documents/Data/Deepak's historical data/rice",
        Area[which(Area > 3)] <- NA, Area[which(Area > 2)] <- NA)  #rice can have 3 growing seasons

    # estimate area of grid cells.
    test <- raster("/Users/ziamehrabi/Documents/Data/Deepak's historical data/soybean/Soybean_1961_Area_ver8.nc")  #template
    area <- area(test)
    ar.ha <- (values(area) * 100)  #convert from km2 to ha
    Prod <- (Area * Yield * ar.ha * 1000)  #get production per cell, in kg

    # save file to RDS
    path2 = paste(newfolder, crops[i], sep = "")
    saveRDS(object = Prod, file = paste(path2, "production.rds", sep = ""))

    # remove files to free up ram
    rm(Area)
    rm(Yield)
    rm(Prod)
}
```

3.1.3 Upscaling the data

Next we upscale the data, and reproject it into Eckert's Equal Area. The final product is production for each crop in $100km^2$ grid cells across the earth, for the years 1961-2008. These are the units of analysis for the manuscript. The code below runs for each input `.rds` file created in the preceding section, and creates a new `.rds` file with the aggregated and reprojected data.

```
rm(list = ls())  #clear space
library(ncdf4)  #load libaries
library(raster)
library(plyr)

newfolder <- "/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/Deepak's historical data/processed"  #folder for output

crops <- list("soybean", "maize", "rice", "wheat")

for (i in 1:length(crops)) {
    crop <- crops[i]
    path = (paste(crop, "production.rds", sep = ""))
    Prod <- readRDS(path)

    # read back in production file to get co-ordinates and append
    names <- colnames(Prod)
    test2 <- nc_open("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/
                Deepak's historical data/soybean/Soybean_1961_Area_ver8.nc")  #get template
    latlon <- expand.grid(lon = ncvar_get(test2, "longitude"), lat = ncvar_get(test2,
        "latitude"))  #get latlong of original
    Prod <- cbind(latlon, Prod)  #bind latlon to production dataset
    nc_close(test2)

    # create a spatial data frame of the original data (for converting to
    # raster) and raster for aggregating up cells.
    coordinates(Prod) <- ~lat + lon  #make spatial
    rasterD <- raster(ncol = 4320, nrow = 2160)  #set original extent and dimensions

    # aggregate then reproject (quicker, less accurate).
    v = list(ncol(Prod))
    for (j in 1:ncol(Prod)) {
        values(rasterD) <- Prod[[j]]

        values(rasterD) <- ifelse(is.na(values(rasterD)), 0, values(rasterD))  #setting NA to zero (required)
        Raster.ag <- aggregate(rasterD, fact = 12, fun = sum, na.rm = T)  #aggregate to 300km2 grid
        Raster.D.equal <- projectRaster(Raster.ag, res = c(1e+05, 1e+05), crs = "+proj=eck4 +datum=WGS84 +ellps=WGS84 +towgs84=0
            method = "ngb", over = T)  #convert to equal area

        values(Raster.D.equal) <- ifelse(values(Raster.D.equal) == 0, NA, values(Raster.D.equal))  #reset zeros to NA.

        v[[j]] <- values(Raster.D.equal)
    }

    # add col names and convert back to df
    v.df <- data.frame(matrix(unlist(v), ncol = 48))
    colnames(v.df) <- names
    scaled.prod <- cbind(v.df, coordinates(Raster.D.equal))
    saveRDS(object = scaled.prod, file = paste(crop, "production100km2.rds",
        sep = ""))
}
```

3.2 Climate data

The aim of this section is to create time series estimates of growing season temperature and precipitation for major crops – maize, soy wheat and rice. We do this by masking climate data by the growing season and locations of crops respresented in our production data.

3.2.1 Spatial masks

Here we expand a function to create a spatial mask for any set of crops from the Monfreda et al. cropland data [5]. Data on the fractional harvested area of crops will be used as a spatial mask. The loop below takes a 0.083d raster, aggregates it, and saves the data as a data frame. It relies on the raster package.

```
rm(list = ls())  #clear space
library(raster)  #load libaries
crops <- c("maize", "rice", "soybean", "wheat")
for (i in 1:length(crops)) {
    file <- paste("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/HarvestedAreaYield175Crops_Geotiff/
            HarvestedAreaYield175Crops_Geotiff/HarvestedAreaYield175Crops_Geotiff/",
        crops[i], "_HarvAreaYield_Geotiff/", crops[i], "_HarvestedAreaFraction.tif",
        sep = "")
    raster.ag <- aggregate(raster(file), fact = 6, fun = mean)
    data <- data.frame(data = values(raster.ag), lon = values(init(raster.ag,
        "x")), lat = values(init(raster.ag, "y")))
    to <- ("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/spatial/")
    saveRDS(object = data, file = (paste(to, crops[i], "mask.rds", sep = "")))
}
```

3.2.2 Temporal masks

The Sacks et al. [9] data is a crop calender data set. It includes planting dates and harvest dates for a range of crops. There are some gaps and caveats with the data, full details are on the download site (https://nelson.wisc.edu/sage/data-and-models/crop-calendar-dataset/index.php). Most notably the data is not suitable for some crops due to poor coverage/ representation. For main crops analyzed here coverage is not bad (see VII. Data coverage section in README file: https://nelson.wisc.edu/sage/data-and-models/crop-calendar-dataset/crop-readme.txt).

I am working with the "filled" , 0.5 degree resolution product. First we import the data, and extract the start date of harvest and the total number of growing days for each grid on the earth. Next we use the start date, and the total number of days of growing season for the crop to create a binary matrix identifying the whether or not a crops growing season is "on" or "off" for each of the 365 days for each grid cell of the earth. The output file represents a 365 *n column matrix, with each column representing the days, and n representing the number of grid cells. In each cell of the array are either 1 or 0's representing whether or not that day includes the growing season for that crop.

```
library(ncdf4)  #load additional libraries
library(progress)
library(pbapply)

crops <- c("Maize", "Rice", "Soybeans", "Wheat", "Maize.2", "Rice.2", "Wheat.Winter")
for (i in 1:length(crops)) {
    file <- paste("Sync/academic/projects/UBCload/Data/Sacks/
            ALL_CROPS_netCDF_0.5deg_filled/",
        crops[i], ".crop.calendar.fill.nc", sep = "")

    mask.t <- nc_open(file)  ##import the data for the crop
    start <- c(ncvar_get(mask.t, "plant.start"))  #get the variables of interest
    tot <- c(ncvar_get(mask.t, "tot.days"))
    nc_close(mask.t)  #close ncfile

    unique(start)  #check start and total number of days the growing season
    unique(tot)

    ## create binary matrix to id if growing is on/off for each day of the year
    start[is.na(start)] <- 0  #convert NAs to zero so seq function works
    tot[is.na(tot)] <- 0

    # get dates by start day number and a given origin date. I chose 2001 as, it
    # is not a leap year
    start.date <- (pblapply(start, function(x) as.Date(x, origin = "2000-12-31")))

    # Then get the sequence of dates based on the number of days the season runs
    # over.
    date.seq <- mapply(FUN = function(x, y) as.Date(seq(x, by = "day", length.out = y)),
        x = start.date, y = tot)

    # Map growing days onto a single 365 day year.
    daysgrowing <- pblapply(date.seq, function(x) as.numeric(strftime(x, format = "%j")))

    # match the values in the vector of growing days in the year to a vector of
    # all possible growing days 1:365
    all.poss <- c(1:365)
    dayincl <- pblapply(daysgrowing, function(x) all.poss %in% x)  #gets the rows

    # Collapse the result into a dataframe, and save it as an RDS file.
    days.incl.df <- data.frame(matrix(unlist(dayincl), nrow = length(tot), byrow = T))

    to <- ("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/temporal/")
    saveRDS(object = days.incl.df, file = paste(to, crops[i], "t.mask.rds",
        sep = ""))
}
```

We create hybrid temporal masks to, for crops with more than one growing season, to cover all of those days the crop is growing in a location irrespective of cropping cycle. We do this for maize, wheat and rice.

```
crops <- c("soybeans", "maize", "rice", "wheat")
files <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/temporal",
    full.names = T)
newfolder <- "/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/temporal_H/"

for (i in 1:length(crops)) {

    files2 <- grep(crops[i], files, ignore.case = TRUE, value = TRUE)  #pull of files with crop
    get.dat <- (lapply(files2, function(x) (readRDS(x))))
    d.all <- Reduce("+", get.dat)
    d.all <- ifelse(d.all == 0, F, T)
    saveRDS(object = d.all, file = paste(newfolder, crops[i], "Hy.mask.rds",
        sep = ""))  #save
}
```

3.2.3 Temperature and precipitation

In this section we highlight the climate data being used for this project, and process it. We are going to use the gridded data from the University of Deleware [11]. This data runs from 1900 to 2014 end. This data is on a monthly time step, so we would need to keep 12*54 = 648 columns if for example we wished to represent from 1961 to the end of 2014.

3.2.4 Rotate data

This data is 0.5 degree spatial point data, that is lat lon, but with longitudes that run from 0-360 rather than -180/+180. So we need to rasterize it and rotate it. However, as it is in a common lat/lon projection, we can just import it directly into the raster package, which is equivalent to truncating the spatial points. Once we have rasterized and rotated these data, we split the data into 12 month chunks. We then save the output to file, as an .rds object. This includes the monthly mean temperature and cumulative monthly precipitation from 1961 -2014 for the globe. In a resolution and CRS that matches that of the filters processed in the previous section.

The below script reads in the files, grabs the most recent years, rotates the data and saves to file. In this case we process a temp and a ppt data for analysis. We save two lists, with each element of the list representing the surface values of the variable for a given year.

```
rm(list = ls())#clear space
library(raster) #load libraries

files<-c("Sync/academic/projects/UBCload/Data/Del/air.mon.mean.v401.nc",

         "Sync/academic/projects/UBCload/Data/Del/precip.mon.total.v401.nc")

type<-c("tempmeans", "pptcum")
for(i in 1:length(files)){
#get the data
stack.del<-stack(files[i])

#trim to selection.
# str(stack.del[[733]]) #this is 1961,1,1
# str(stack.del[[1380]])  #this is "X2014.12.01
stack.del<-stack.del[[733:1380]]

#rotate the output
stack.del.r<-rotate(stack.del)

#plot(stack.del.r[[1]]) #check it looks good

#Once we have rasterized and rotated it, we split the data into 12 month chunks.
data.rotated<-as.data.frame(matrix(values(stack.del.r), nrow= ncell(stack.del.r), ncol=nlayers(stack.del.r)))

#split out the matrix by every 12 months - use split.default to run along cols.
data.rot.list<-split.default(data.rotated, ceiling(seq_along(data.rotated)/12))

#str(data.rot.list)#check - should be a list of 40 years,of 12 elements each!

#We then save the output to file
saveRDS(object = data.rot.list, file = paste("Sync/academic/projects/UBCload/Data/Del/rot.", type[i],".rds", sep=""))
}
```

3.2.5 Extract climate data by masks

In this section we filter the climate data by the crop and temporal masks. We run this over two climatic variables, and for all of the crops. We note that there are more temporal masks than spatial masks – due to the multiple cropping seasons for crops, and so we create climate products for each of these. Below the code runs for the hybrid case (e.g. with combined crop calenders, but can also be run for the single case, with individual crop calenders for each crop's multiple croppings in a year, e.g. winter vs. spring wheat).

```
rm(list = ls())  #clear space
library(raster)  #load libraries
library(rgdal)
# define the input files
clim.files <- c("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/Del/rot.pptcum.rds",
    "/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/Del/rot.tempmeans.rds")
clim.short <- c("ppt", "temp")
spatial.masks <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/spatial",
    full.names = T)  #hybrid
temp.masks <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/temporal_H",
    full.names = T)
rast <- raster(ncol = 720, nrow = 360, crs = "+proj=longat +ellps=WGS84 +datum=WGS84 +towgs84=0,0,0",
    extent(-180, 180, -90, 90))  #set up a raster grid, this is wgs84.
temp.masks.short <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/temporal_H")  #hybrid

for (i in 1:length(clim.files)) {
    # run through all climate files and all crop calenders
    for (j in 1:length(temp.masks)) {


        spatial <- grep(substr(temp.masks.short[j], 1, 4), spatial.masks, ignore.case = TRUE,
            value = TRUE)  #grab the spatial mask that matches the temporal mask
        spatial <- readRDS(spatial)  #read int he spatial mask
        spatial <- ifelse(spatial$data > 0.01, 1, NA)  #set the spatial data to equal 1 and NAs

        # make the daily occurence in the year for growing season equal to 1, and
        # all other equl to NA:
        temporal <- readRDS(temp.masks[j])
        temporal <- data.frame(ifelse(temporal == T, 1, NA))

        # convert temporal mask to monthly. The current mask is by day of the year,
        # but Deleware is by month. therefore we identify any month with a day
        # within the growing season as a 'growing season' month:
        days <- as.Date(seq(as.Date(1, origin = "2000-12-31"), by = "day", length.out = 365))  #get days of each month
        days.s <- substr(days, 6, 7)  #get the month of those days
        # plyr::count(days.s)$freq #get the frequency
        temp.sp <- split.default(temporal, days.s)  #split out cols by the days factor
        temp.split.binary <- lapply(temp.sp, function(x) (rowSums(is.na(x)) ==
            ncol(x)))  #get rows that all = NA
        tspbinall <- do.call(cbind, temp.split.binary)  #cbind
        tspbinall <- ifelse(tspbinall == T, NA, 1)  #set all rows with NA to NA and others to 1

        # Then read in all of the temperature records for each year, filter each
        # year by the spatial mask, and the temporal mask imported above.  Get mean
        # monthly growing season temperature.
        clim <- readRDS(clim.files[i])
        clim.brick <- NULL
        for (k in 1:length(clim)) {
            climfile <- clim[[k]]
            climfile <- spatial * climfile  #gives spatial mask for each crop
            climfile <- climfile * tspbinall  #multplies the spatial mask by the temporal mask
            year.med <- apply(climfile, 1, function(x) mean(x, na.rm = T))  #get row mean
            values(rast) <- year.med  # add vaues to a raster
            rm(climfile)  #free ram
            clim.brick <- c(clim.brick, rast)
        }
        ## save the output for this temporal mask[j] and climate product[i] to file
        saveRDS(clim.brick, paste("/Users/pumpkinjr/Sync/academic/projects/
                          UBCload/Data/masks2/mask_Del/hybrid/",
            clim.short[i], temp.masks.short[j], sep = ""))
    }
}
```

3.2.6 Upscaling the data

To undertake the analysis on the scale of the production data we aggregate the climate variables, and we also convert the products into equal area. When we do this maintain mean estimates of temperature, and cumulative sums of precipitation.

```
rm(list = ls())  #clear space
library(ncdf4)  #load libaries
library(raster)
library(plyr)
library(pbapply)

newfolder <- "/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/mask_Del_100km_eck4/hybrid/"  #folder for output
files.short <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/mask_Del/hybrid/")
files <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/mask_Del/hybrid/",
    full.names = T)

pb <- progress_bar$new(total = length(files))  #set up progress bar
for (i in 1:length(files)) {
    # run through crop calender climate files
    t <- readRDS(files[i])  #read file
    b <- stack(t)  #make stack of years
    values(b) <- ifelse(is.na(values(b)), 0, values(b))  #setting NA to zero (required)
    x <- if (length(grep("temp", files[i])) == 1) {
        mean
    } else {
        sum
    }  #choose function type for aggregation.
    b.ag <- aggregate(b, 2, fun = x, na.rm = T)  #scale up
    b.ag.eck4 <- projectRaster(b.ag, res = c(1e+05, 1e+05), crs = "+proj=eck4 +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0",
        method = "ngb", over = T)  #convert to equal area
    values(b.ag.eck4) <- ifelse(values(b.ag.eck4) == 0, NA, values(b.ag.eck4))  #reset zeros to NA.
    dat <- cbind(as.data.frame(values(b.ag.eck4))[, 1:48], coordinates(b.ag.eck4))  #grab 1961-2008, append co-ords
    saveRDS(object = dat, file = paste(newfolder, files.short[i], "100km2.rds",
        sep = ""))  #save
    pb$tick()
    Sys.sleep(1/100)
}
```

4 Analysis

In this section we perform the analysis presented in the manuscript. First we plot the spatial differences in production for each of the crops for context. Second, we map out the local contributions to global variation in crop production for the time period of our study. Third, we break the time series down to assess how local stability or synchrony has changed over time, and how much each has contributed to global instability for the four crops used in our analysis over the period 1961-2008. Fourth, we take a look at between crop synchrony to identify if there is evidence for compensatory dynamics between crops for calorie production. Fifth we undertake a thought experiment to see how large production deficits would have been in 2008, if all production cells across the world were synchronized. We then see what how much of an we would have to implement different mitigation strategies to offset this production loss. Finally, we take a look at synchrony in climate for major crop growing locations and asesss if this shows the same pattern as that for crop production.

4.1 Spatial differences in production over 1961-2008

First we plot out the shifts in spatial patterns and concentration of each of the crop over 1961-2008 to provide the context for the stability analysis that follows. We read in each of the two years of the aggregated data.

```
rm(list = ls())  #clear space
maize <- readRDS("maizeproduction100km2.rds")
rice <- readRDS("riceproduction100km2.rds")
soybean <- readRDS("soybeanproduction100km2.rds")
wheat <- readRDS("wheatproduction100km2.rds")
```

Below is a function to plot each dataset in turn, with a specifies break for that particular crop.

```
library(RColorBrewer)  #load libraries
library(raster)
library(maptools)
map.scenarios <- function(data, prod, title, breaks.crop) {
    data(wrld_simpl)
    df <- data.frame(data$x, data$y, prod)
    dfr <- rasterFromXYZ(df)  #Convert first two columns as lon-lat and third as value
    crs(dfr) <- "+proj=eck4 +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"
    colfunc <- colorRampPalette(brewer.pal(7, "Greens"))  #set the color data(wrld_simpl) #get world map data
    values(dfr) <- (100 * (values(dfr))/sum((values(dfr)), na.rm = T))
    wrld <- spTransform(wrld_simpl, crs(dfr))
    plot(wrld, col = "gray", border = "gray", lwd = 0.2)
    plot(dfr, col = colfunc(5), add = TRUE, legend.width = 1, legend.shrink = 0.5,
        legend.args = list(text = "", side = 4, cex = 0.8, line = 2), title(main = title),
        breaks = breaks.crop)
    plot(wrld, col = "transparent", border = "black", lwd = 0.2, add = T)
}
```

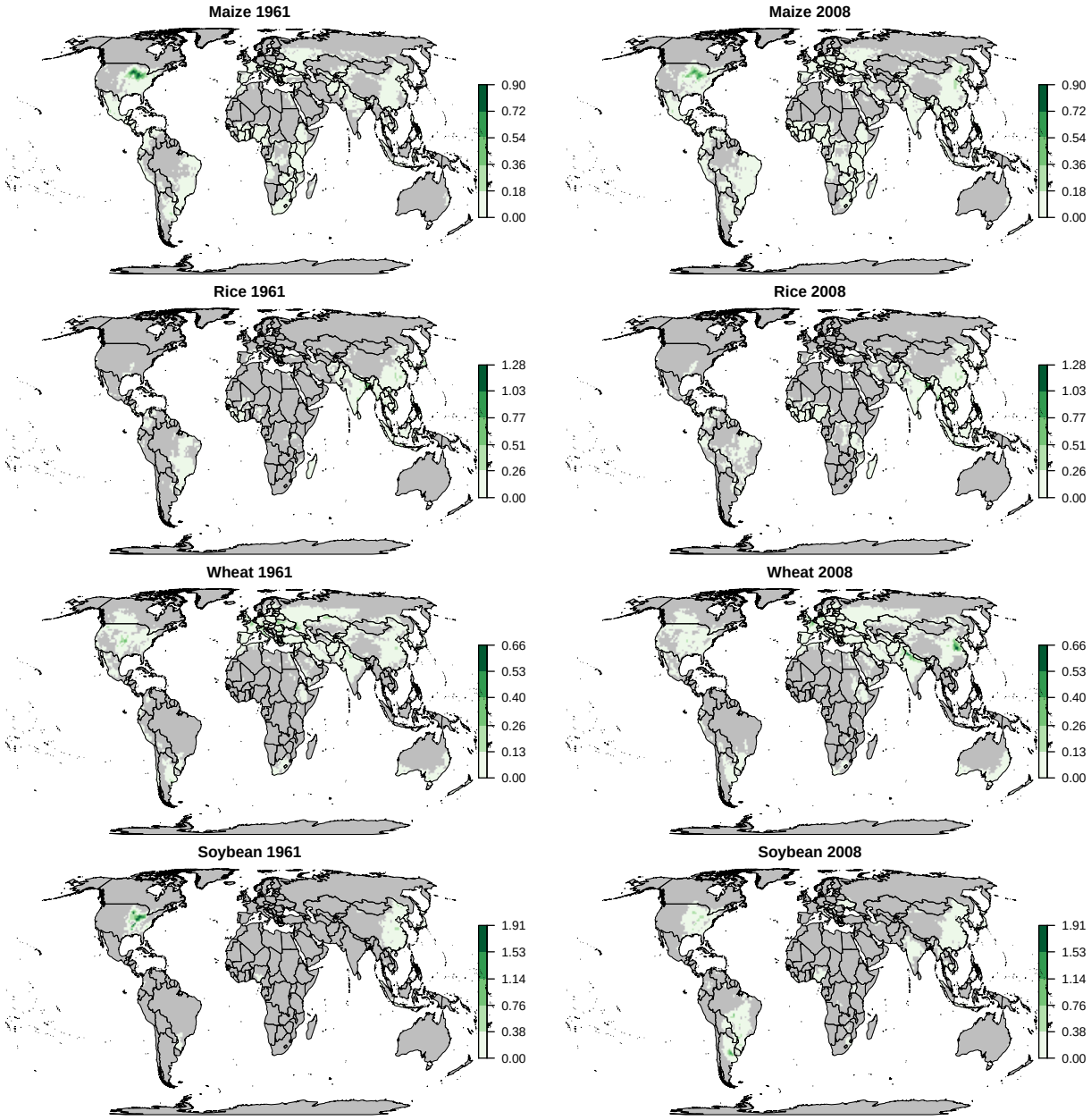
Then we define the maximum percentage contribution to production that any cell makes over the two time steps for each crop type. And use this to define the breaks for the plotting.

```
rice.max <- max(max(rice$Rice_1961_Area_ver8.nc/sum(rice$Rice_1961_Area_ver8.nc,
    na.rm = T), na.rm = T), max(rice$Rice_2008_Area_ver8.nc/sum(rice$Rice_2008_Area_ver8.nc,
    na.rm = T), na.rm = T)) * 100

maize.max <- max(max(maize$Maize_1961_Area_ver8.nc/sum(maize$Maize_1961_Area_ver8.nc,
    na.rm = T), na.rm = T), max(maize$Maize_2008_Area_ver8.nc/sum(maize$Maize_2008_Area_ver8.nc,
    na.rm = T), na.rm = T)) * 100

soy.max <- max(max(soybean$Soybean_1961_Area_ver8.nc/sum(soybean$Soybean_1961_Area_ver8.nc,
    na.rm = T), na.rm = T), max(soybean$Soybean_2008_Area_ver8.nc/sum(soybean$Soybean_2008_Area_ver8.nc,
    na.rm = T), na.rm = T)) * 100

wheat.max <- max(max(wheat$Wheat_1961_Area_ver8.nc/sum(wheat$Wheat_1961_Area_ver8.nc,
    na.rm = T), na.rm = T), max(wheat$Wheat_2008_Area_ver8.nc/sum(wheat$Wheat_2008_Area_ver8.nc,
    na.rm = T), na.rm = T)) * 100

breaks.maize <- round(seq(0, maize.max + 0.01, length.out = 6), digits = 2)
breaks.rice <- round(seq(0, rice.max + 0.01, length.out = 6), digits = 2)
breaks.wheat <- round(seq(0, wheat.max + 0.01, length.out = 6), digits = 2)
breaks.soy <- round(seq(0, soy.max + 0.01, length.out = 6), digits = 2)
```

4.1.1 Supplementary Figure 1

And finally we plot each of the layer in a single figure. As can be shown below some crops like wheat have more of a spatially dispersed distribution, than other crops such as soy or maize which has production that is more spatially concentrated on a global level. We may expect this to influence the dynamics of stability for each crop, with particularly productive locations contributing to a greater variance in production also.

```
par(mfrow = c(4,2),
      oma = c(0,0,0,0) ,
          mar = c(0,0,1,0))

map.scenarios(maize, maize$Maize_1961_Area_ver8.nc, "Maize 1961", breaks.maize)
map.scenarios(maize, maize$Maize_2008_Area_ver8.nc, "Maize 2008", breaks.maize)
map.scenarios(rice, rice$Rice_1961_Area_ver8.nc, "Rice 1961",breaks.rice)
map.scenarios(rice, rice$Rice_2008_Area_ver8.nc, "Rice 2008",breaks.rice)
map.scenarios(wheat, wheat$Wheat_1961_Area_ver8.nc, "Wheat 1961",breaks.wheat)
map.scenarios(wheat, wheat$Wheat_2008_Area_ver8.nc, "Wheat 2008", breaks.wheat)
map.scenarios(soybean, soybean$Soybean_1961_Area_ver8.nc, "Soybean 1961",breaks.soy)
map.scenarios(soybean, soybean$Soybean_2008_Area_ver8.nc, "Soybean 2008", breaks.soy)
```



Supplementary Figure 1: Local contributions to global crop production in 1961 and 2008. Each pixel represents a 100 km x 100 km grid cells percent contribution to global crop production for two time points

4.2 Time detrending

Throughout this manuscript we will be using local regression (loess) to detrend the production time series. This is because we are primarily interested in the year-year variation in production, which would otherwise be swamped by longer term trends, such as those due to technological change e.g. increased input use or total factor productivity. Below we write a function to check how different span parameters influence the fit.

```
rm(list = ls())  #clear space
library(tidyr)  #load libraries
library(ggplot2)
library(cowplot)

check.span <- function(dat, span.p, plot.n) {

    # takes:production data, a span length for loess, number of plots to return
    # returns: plots of fits for random selection of locations

    cord <- subset(dat, , c(x, y))  #get co-ordinates
    NAs <- apply(dat, 1, function(x) sum(is.na(x)))  #count NAs
    w.compl <- subset(dat, NAs < 48)  #remove cases with 48 NAs
    w.compl <- subset(w.compl, , -c(x, y))  #remove co-ordinates
    w.compl[is.na(w.compl)] <- 0  #replace NA with zeros (these are zero production years)
    year <- 1961:2008  #set year

    w.mod <- apply(w.compl, 1, function(x) loess(x ~ year, span = span.p))  #detrend
    w.fitted <- as.data.frame(t(data.frame((lapply(w.mod, function(x) x$fitted)))))  #get fittted
    w.resid <- as.data.frame(t(data.frame((lapply(w.mod, function(x) x$residuals)))))  #get resid

    w.fitted$id <- w.resid$id <- w.compl$id <- paste(as.character(1:nrow(w.compl)))  #add id
    colnames(w.fitted) <- colnames(w.resid) <- colnames(w.compl) <- c(1961:2008,
        "id")  #add colnames

    fits.res <- gather(w.resid, year, res, "1961":"2008")
    fits.l <- gather(w.fitted, year, fit, "1961":"2008")
    obs.l <- gather(w.compl, year, o, "1961":"2008")

    datall <- data.frame(obs = obs.l$o, year = as.numeric(fits.res$year), fit = fits.l$fit,
        res = fits.res$res, id = fits.res$id)

    # randomly sample a selection
    set.seed(2)  # set seed for reproducibility
    locs <- unique(datall$id)
    random.no <- runif(length(locs), min = 0, max = 1)
    get.id <- data.frame(locs, random.no)[order(random.no), ]$locs[1:plot.n]
    trimmed <- subset(datall, datall$id %in% get.id)

    # plot 1 - fit
    result1 <- ggplot(trimmed, aes(year, obs, by = id)) + geom_point(pch = 19) +
        geom_line(aes(year, fit)) + # geom_hline(yintercept = 0)+
    facet_wrap(~id, scale = "free") + theme_bw()

    # #plot 2 - residual plot result2<- ggplot( trimmed , aes(fit,(res), by=id)
    # )+ geom_point(pch=19)+ facet_wrap(~id, scale='free')+ theme_bw()

    return(result1)
}
```

We used the above function to check different loess span values for multiple random samples of production time series, across the four different crops (not shown).

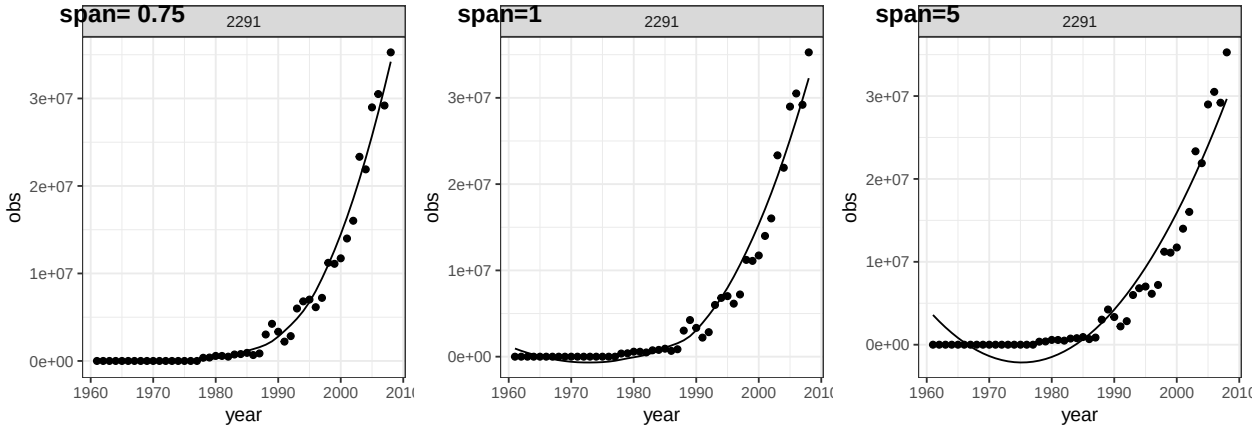
```
maize <- readRDS("maizeproduction100km2.rds")
rice <- readRDS("riceproduction100km2.rds")
wheat <- readRDS("wheatproduction100km2.rds")
soy <- readRDS("soybeanproduction100km2.rds")

check.span(maize, 0.75, 20)  #check default span
check.span(rice, 0.75, 20)
check.span(wheat, 0.75, 20)
check.span(soy, 0.75, 20)

check.span(maize, 1, 20)  #check span of 1
check.span(rice, 1, 20)
check.span(wheat, 1, 20)
check.span(soy, 1, 20)

check.span(maize, 5, 20)  #check span of 5
check.span(rice, 5, 20)
check.span(wheat, 5, 20)
check.span(soy, 5, 20)
```

Ideally, we might like to increase the span from the default of 0.75 to represent longer term trends in production. However, we chose the default smoothing parameter because larger spans introduced artifacts into the results. This is shown by the figure below. The default span of 0.75 provides the best approximation of the observed values. Increasing the span, even to a span of 1, leads to unnecessary curvature in the predicted production in early years in this case. As the span increases so does the magnitude of the non-sense values in the prediction.

```
soy <- readRDS("soybeanproduction100km2.rds")
s2.3 <- check.span(soy, 0.75, 1)
s1 <- check.span(soy, 1, 1)
s5 <- check.span(soy, 5, 1)
loess.span.check <- plot_grid(s2.3, s1, s5, align = c("v", "h"), ncol = 3, labels = c("span= 0.75",
    "span=1", "span=5"))
```

4.2.1 Supplementary Figure 2

```
loess.span.check
```



Supplementary Figure 2: Different loess span options example. For functions where production increases throughout the time series from a low or zero production, increasing the span of the loess results in undesirable predicted production values.

4.3 Estimate local contributions

4.3.1 Variance contribution index

In this subsection we map out the local contributions to global variance of crop production for four major crop types. These local contributions are due to both local variation in production at the grid cell level, and the correlation in production between different grid cells across the world. First we detrend the production time series (to ensure the contributions reflect year-year variation, which would otherwise be swamped by mean technology lead increases in production over 1961-2008), then we compute the following index for each focal grid cell on the planet for each crop:

$I_{z \in n}$ = (1-($\sigma_G{}^2_{n-z} \div \sigma^2_G$))*100

Where σ^2_G is the global variance and $\sigma_G{}^2_{n-z}$ is the global variance when a given grid cell z is removed from the total number of producing grid cells n. We computed this index independently for each of the four crops used in our analysis.

Formally, σ^2_G is equal to $\sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j}$, where i and j are the respective rows and columns of a symmetric variance-covariance matrix X. The diagonal elements of X represent the temporal variance in production for a given grid cell over 1961-2008, and the off-diagonal elements of of X represent the temporal covariances in production between grid cells over the same time period. Removing z can either increase or decrease σ^2_G. As such, I_z represents the stabilizing or destabilizing effects of grid cell z on global crop production over the time period, resulting from inter-annual variation. I_z is in units of percent, where negative values represent a grid cells percent deflation of variance, and positive values represent a percent inflation of variance, of global production over the time period 1961-2008.

In the code that follows we first detrend the time series of production, then compute I_z for each n grid cell. We do this independently for each crop. The output is stored as a list named "all.crop.var". Each element of the list is a crop, and the numerical values in that list element represent the destabilization index. Note that the co-ordinates of the grid cell have been removed from this final product, we will need to add them in back later.

```
rm(list = ls())  #clear space
library(raster)  #load libraries
library(sp)
library(maptools)
library(grid)
library(rgdal)
library(ggplot2)

crops <- list("maize", "rice", "soybean", "wheat")
year <- 1961:2008
all.crop.var = list()

for (i in 1:length(crops)) {
    crop <- crops[i]  #assign crop
    path = (paste(crop, "production100km2.rds", sep = ""))  #create path
    w <- readRDS(path)  #read in
    D <- w[, -which(names(w) %in% c("y", "x"))]  #remove co-ords.

    D[is.na(D)] <- 0  # replace NA with zeros, otherwise we get NAs in the denominator.
    c = colSums(D, na.rm = T)  #global production trends for whole dataset
    Dn <- t(apply(D, (1), function(x) c - x))  #global production minus contribution of grid cell

    # get variance of detrended production for data with grid cells removed
    Dn2 <- data.frame(t(apply(Dn, 1, function(x) resid(loess(x ~ year)))))  #use loess for detrending

    var.detrend <- apply(Dn2, 1, var)  #get variance of each row
    var.dc <- var(resid(loess(c ~ year)))

    # get ratio, >1 then detsabilizing, < 1 then stabilizing is global variance
    # more or less than with the cell?
    v.ratio <- var.detrend/var.dc

    # remove 1's cases where grid cell prod is 0 and so the ratio is exactly
    # equal to 1.
    v.ratio.new <- replace(v.ratio, v.ratio == 1, NA)
    v.ratio.new <- (round((1 - v.ratio.new) * 100, digits = 1))  #change to percentage

    # bind alltogether in dataframe + save
    name <- crop[[1]]
    all.crop.var[[name]] <- v.ratio.new

}
```

4.3.2 Summary of the local contributions

In this section we quickly summarise the distributions of the variance contribution index for each crop. Expanding this summary we will see the maximum variance inflating and deflating locations differ for each crop and so does the mean value. Most locations are all close to zero in their values, which means they have no practically significant impact on year-year variation in global crop production.

```
summarydat <- lapply(all.crop.var, function(x) quantile(x, probs = seq(0, 1,
    0.01), na.rm = T))
```

4.3.3 Plot local contributions

Next we plot the local contributions to global variance for each crop. We will use the co-ordinates called in the loop in the previous last section to create a common plotting space with the correct dimensions, projection and co-ordinate system. Once we have this we will assign the data to this common template, and plot out the data on global maps.

```
data <- data.frame(w$x, w$y, v.ratio.new)  #fill results into a matrix
coordinates(data) <- ~w.x + w.y
gridded(data) <- TRUE
raster.w <- raster(data)
raster.x <- raster(res = c(1e+05, 1e+05), crs = "+proj=eck4 +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")
projection(raster.w) = projection(raster.x)
```

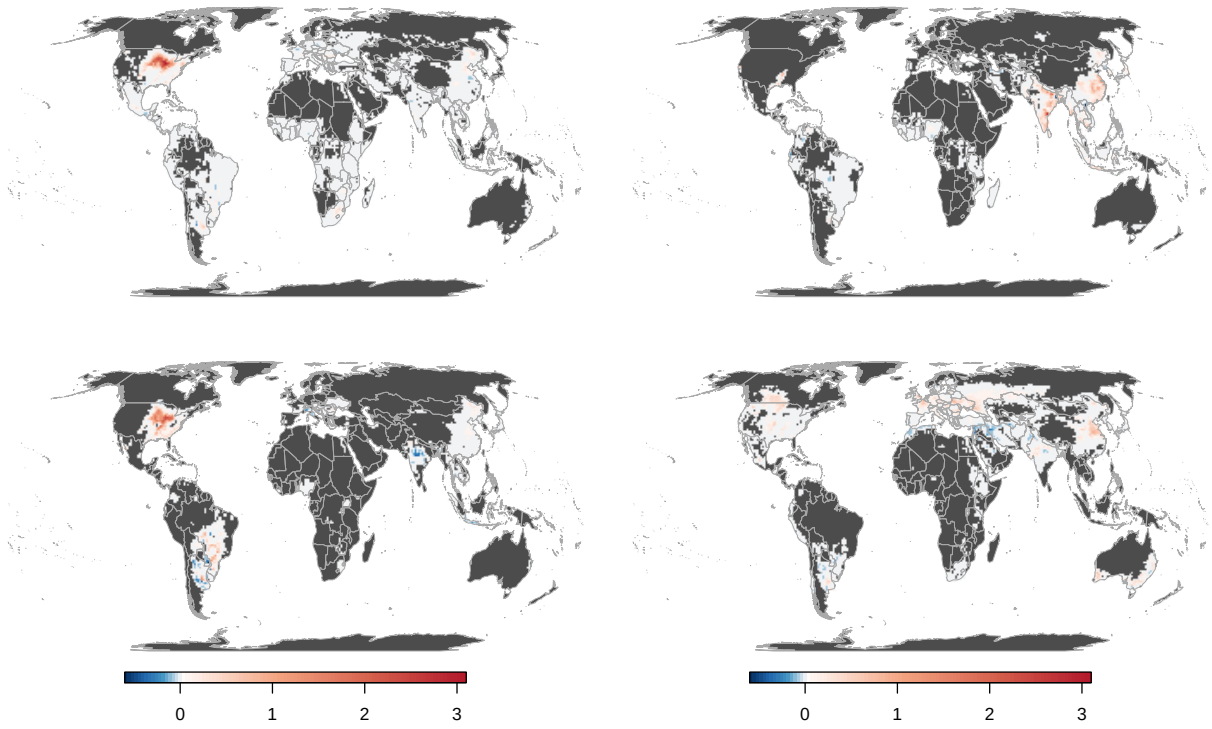
Finally, we plot out the maps. First we set up the color schemes, and the min and maximum values for the common legend for plotting, and the breaks for the colors.

```
# get min and max across the four plots for common legend
min_ = min(c(all.crop.var[[1]], all.crop.var[[2]], all.crop.var[[3]], all.crop.var[[3]]),
    na.rm = T)
max_ = max(c(all.crop.var[[1]], all.crop.var[[2]], all.crop.var[[3]], all.crop.var[[3]]),
    na.rm = T)

# assign breaks
breaks <- c(seq(min_, 0, length.out = 100))
breaks2 <- c(seq(1e-06, max_, length.out = 100))

## blue red
color.palette = (colorRampPalette(c("#053061", "#2166ac", "#4393c3", "#f7f7f7"))(length(breaks2)))  ##with blue
color.palette2 = (colorRampPalette(c("#fffbf9", "#f4a582", "#d6604d", "#b2182b"))(length(breaks)))  ##red side

# join them together
breaks.all <- c(breaks2, breaks)
colors.all <- c(color.palette, color.palette2)
```

4.3.4 Supplementary Figure 3

Then we run a loop to plot each crop in turn. We first set the plotting frame in to be 2x2 units using the graphical parameters function par(), and assign each element of the "all.crop.var" list to the plotting frame. Some manipulation of the plotting area using par() was undertaken to ensure the plot is compact for publication. We run the loop for the first and second rows of the panel, with the legend included in the second row to make the presentation neater.

```
data(wrld_simpl) #get world map data

par(mfrow = c(2,2),#set up plottting area
        oma = c(26,0,0,0) + 0.1,
          mar = c(0,0,0,0) + 0.1)


for(i in 1:2){
all.crop.var
values(raster.w)<-all.crop.var[[i]]#assign values to raster
wrld <- spTransform(wrld_simpl, crs(raster.x))
plot(wrld, ylim=c(-99910,99820),  col ="#4D4C4D",border=NA, bg="white" )
plot(raster.w,col = colors.all, cex=1, breaks=breaks.all, add=TRUE, legend=F)
 plot(wrld, ylim=c(-99910,99820), col='transparent', border="dark gray", add=TRUE,lwd=0.2 ) #add transparent borders
}

for(i in 3:4){
values(raster.w)<-all.crop.var[[i]]#assign values to raster
wrld <- spTransform(wrld_simpl, crs(raster.x))#plot out
plot(wrld, ylim=c(-99910,99820),  col ="#4D4C4D",border=NA, bg="white" )
plot(raster.w,col = colors.all, cex=1, breaks=breaks.all, add=TRUE, horizontal=TRUE,
     legend.mar =0, legend.width = 1.5, legend.shrink=0.55,legend.args=list(text='',
                      font=2, line=1, cex=1), axis.args=list(at=pretty(c(round(min_, digits=2),0,round(max_, digits=2))),
                     labels=pretty(c(round(min_, digits=2),0,round(max_, digits=2)))))
 plot(wrld, ylim=c(-99910,99820), col='transparent', border="dark gray", add=TRUE,lwd=0.2 ) #add transparent borders
}
```



Supplementary Figure 3: Local contributions to global variance of crop production over 1961-2008. Each pixel represents a 100 km x 100 km grid cells percent contribution to inter-annual global variance in crop production for the last five decades. Negative values show variance deflating locations, and positive values show variance inflating locations. Crops shown, from top right to bottom left: Maize, Rice, Soy, Wheat

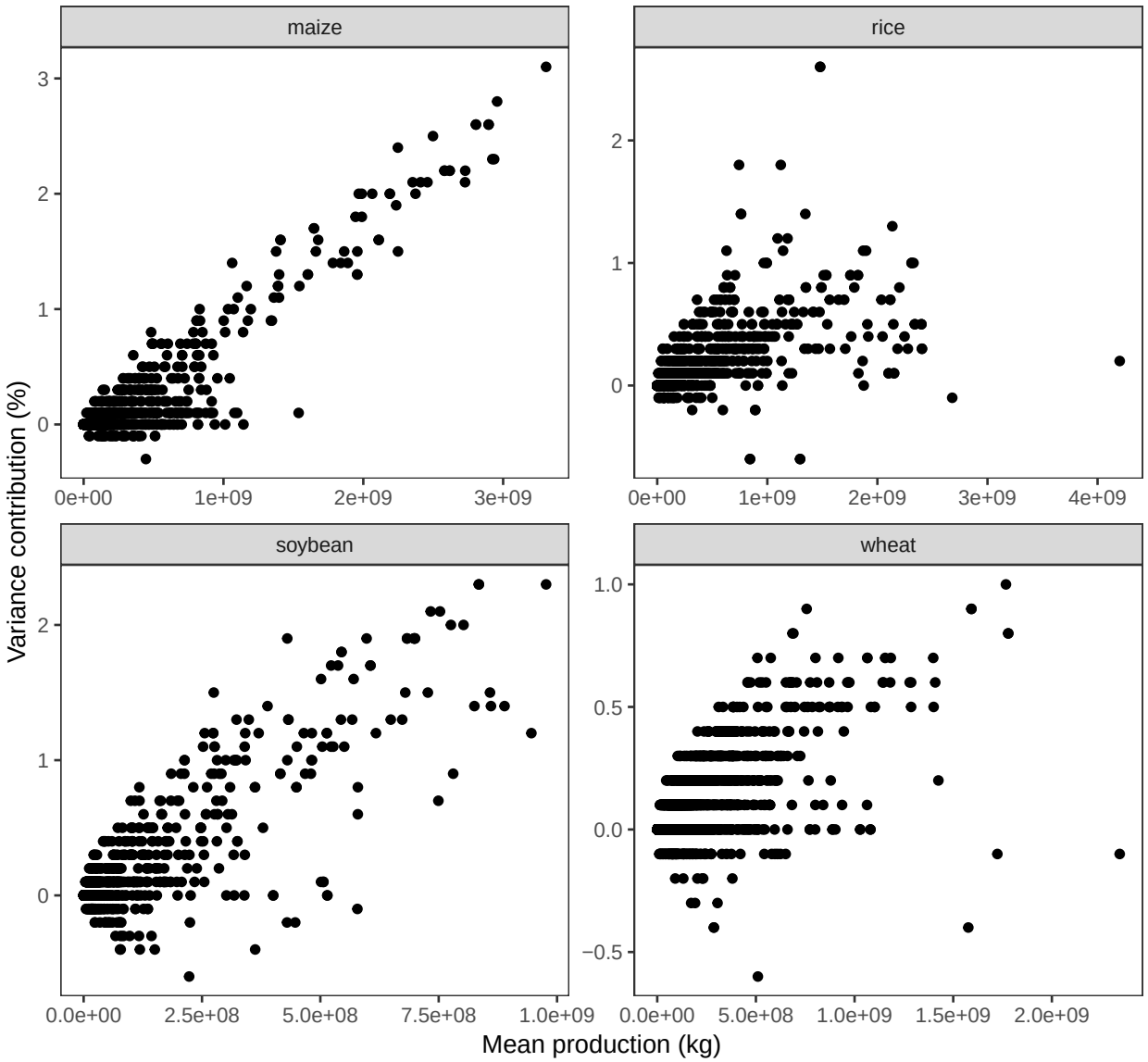
4.3.5 Supplementary Figure 4

In this subsection we plot the mean production of a grid cell against the variance contribution index. This is because we may expect that grid cells that have a large impact on the global variance may also represent cells that contribute disproportionately to the global production.

```
# get mean production by row and by crop
crops <- list("maize", "rice", "soybean", "wheat")
all.mean = data.frame()

for (i in 1:length(crops)) {
    crop <- crops[i]  #assign crop
    path = (paste(crop, "production100km2.rds", sep = ""))  #create path
    w <- readRDS(path)  #read in
    w <- subset(w, , -c(x, y))  #remove co-ordinates
    crop.prod <- rowSums(w, na.rm = T)/length(w)
    crop.prod <- data.frame(crop.id = paste(crop), mean.prod = crop.prod)
    all.mean <- rbind(all.mean, crop.prod)
}


# unlist the variance contributions.
var.cont <- unlist(all.crop.var)

# join the two data files
meanvsvar <- cbind(all.mean, var.cont)
```

Now we plot those relationships in a scatter below. As we can see, for maize and for soybean in particular, cells which produce extremely large amounts of product tend to contribute more to the global variance in production trends. The relationships are not so clear for cells at the lower ends of production for these crops, nor for rice and for wheat.

```
ggplot(meanvsvar, aes(mean.prod, var.cont)) + geom_point() + # stat_smooth(se=F) +
facet_wrap(~crop.id, scales = "free") + xlab("Mean production (kg)") + ylab("Variance contribution (%)") +
    theme_bw() + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    legend.position = "none", legend.key = element_blank())
```



Supplementary Figure 4: Mean versus variance contribution relationships for crop producing locations over 1961-2008

4.4 Decadal analysis

4.4.1 Summary indices

In this subsection we calculate the global instability (CV_G), local instability (CV_L) and synchrony (ϕ), for the four crops within 8 year windows of the 1961-2008 time period. We draw on recent theory developed for scaling stability in productivity in ecology [10]. First we detrend the data (using loess regression to account for non-linearity in production trends), then calculate these summary statistics, before storing them in a common data frame "all.crop" ready for plotting.

We compute the global and local standard deviations as:

$\sigma_G = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j}}$.

and,

$\sigma_L = \sum_{i=1}^{n} \sqrt{x_{i,i}}$

We then compute the global instability and local instability in crop productivity as the global and local coefficients of variation in production for each of the crops at each time window in the analysis:

$CV_G = \sigma_G \div \mu_G$

and,

$CV_L = \sigma_L \div \mu_G$

Where μ_G is the non-deterended mean of global production. Note, that this formulation (with non-detrended productivity data for the mean, and mean detrended data for the standard deviation) overcomes the influence of non-stationarity in the mean on inter-annual variance (i.e. due to technology change), but ensures an informative picture of the relative severity of losses is maintained (e.g. a -50% deviation from the mean in 1961 is much smaller in absolute terms than a -50% deviation in 2008).

Finally we compute a the third diagnostic metric, synchrony:

$\phi = \sigma_G^2 \div (\sum_{i=1}^{n} \sqrt{x_{i,i}})^2$

Where ϕ is the synchrony between the all the producing grid cells in the world for a given crop. This denominator of this ratio, $(\sum_{i=1}^{n} \sqrt{x_{i,i}})^2$, or σ_L^2, is equal to σ_G^2 when all elements of the correlation matrix of producing grid cells (P) have correlation of $\rho = 1$. This index is bounded by 1, complete synchrony and approaches 0, when all the average elements of P equal $-1/(n-1)$, where n is the number of producing grid cells, to give complete asynchrony.This metric is useful because it shows how close we are globally we have been the 'worst case' scenario of complete synchronous production dynamics over the period 1961-2008.

These three quantities are related such that:

$CV_G{}^2 = CV_L{}^2 \cdot \phi$

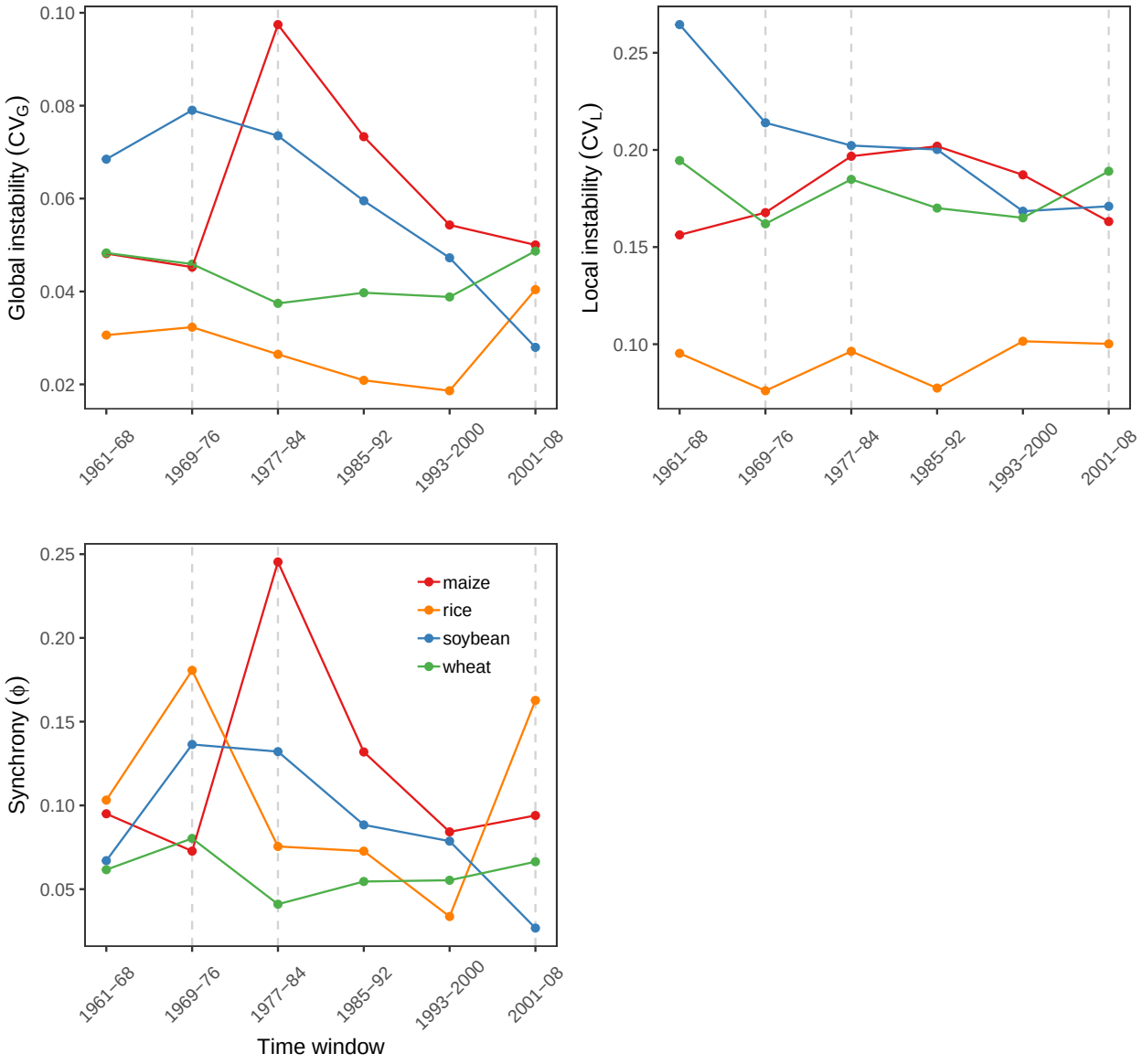
With ϕ acting as a scaling factor that links stability at the local to the global scale [10]

```
rm(list = ls())  #clear space
crops <- list("maize", "rice", "soybean", "wheat")
all.crop = data.frame()

for (i in 1:length(crops)) {
    crop <- crops[i]  #assign crop
    path = (paste(crop, "production100km2.rds", sep = ""))  #create path
    w <- readRDS(path)  #read in
    w <- subset(w, , -c(x, y))  #remove co-ordinates

    w.compl <- w[rowSums(is.na(w)) != ncol(w), ]  #remove complete cases of NA
    w.compl[is.na(w.compl)] <- 0  #replace NA with zeros

    year <- 1961:2008
    w.compl <- w.compl/1e+10  #convert kg to into mega tonnes

    w.compl.dec <- list(w.compl[1:8], w.compl[9:16], w.compl[17:24], w.compl[25:32],
        w.compl[33:40], w.compl[41:48])

    w.compl.dec.sum <- lapply(w.compl.dec, function(x) colSums(x))

    detrend.w <- data.frame(t(apply(w.compl, 1, function(x) resid(loess(x ~
        year)))))  #use loess for detrending

    det.w.dec <- list(detrend.w[1:8], detrend.w[9:16], detrend.w[17:24], detrend.w[25:32],
        detrend.w[33:40], detrend.w[41:48])  #split data into windows

    det.w.dec <- lapply(det.w.dec, t)  #transpose to get the grid cells on the cols

    diff.cov <- lapply(lapply(det.w.dec, data.frame), cov)  #make covariance matrix of the grid cells.
    diag.cov <- lapply(diff.cov, diag)  #get diagonals of covariance matrix = local variance

    sqrt.diag.cov <- lapply(diag.cov, sqrt)  #sqrt of diags of covariance matrix = list of local sds

    diff.var.glob <- unlist(lapply(diff.cov, sum))  #sum the covariance matrix. = global variance
    diff.var.local <- unlist(lapply(diag.cov, sum))  #sum the diagonals = local variance

    diff.sd.global <- sqrt(diff.var.glob)  #global sd
    diff.sd.local <- unlist(lapply(sqrt.diag.cov, sum))  #local sd

    diff.sync <- diff.var.glob/(diff.sd.local^2)  #synchrony

    mean.dec <- unlist(lapply(w.compl.dec.sum, mean))  #non-detrended global means

    CVl <- diff.sd.local/mean.dec  #local instability
    CVg <- diff.sd.global/mean.dec  #global instability

    # bind all together in dataframe + save
    vars <- c(rep("global variance", 6), rep("local variance", 6), rep("synchrony",
        6), rep("CVl", 6), rep("CVg", 6))
    data <- c(diff.var.glob, diff.var.local, diff.sync, CVl, CVg)
    year <- c(rep(1:6, 5))
    cropid <- c(rep(unlist(crop), 30))
    all <- data.frame(vars, data, year, cropid)  #crop specific
    all.crop <- rbind(all.crop, all)
}
saveRDS(all.crop, "allcrop.rds")
```

4.4.2 Supplementary Figure 5

In this subsection we plot the data from the decadal analysis above.

```
library(ggplot2)  #load libraries
library(gridExtra)
library(cowplot)

all.c.split <- split(all.crop, all.crop$vars)  #split buy variable to plot individually


global <- ggplot(all.c.split$CVg, aes(year, (data), colour = cropid)) + geom_vline(xintercept = 2,
    lty = 2, color = "light gray") + geom_vline(xintercept = 3, lty = 2, color = "light gray") +
    geom_vline(xintercept = 6, lty = 2, color = "light gray") + geom_point() +
    geom_line() + theme_bw() + scale_color_manual(values = c("#e41a1c", "#ff7f00",
    "#377eb8", "#4daf4a")) + ylab(expression(Global ~ instability ~ (CV[G]))) +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        legend.position = "none", legend.key = element_blank(), legend.key.size = unit(0.5,
            "cm"), legend.title = element_blank(), axis.text.x = element_text(angle = 45,
            vjust = 0.5)) + xlab("") + scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68",
    `2` = "1969-76", `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08"))

local <- ggplot(all.c.split$CVl, aes(year, (data), colour = cropid)) + geom_vline(xintercept = 2,
    lty = 2, color = "light gray") + geom_vline(xintercept = 3, lty = 2, color = "light gray") +
    geom_vline(xintercept = 6, lty = 2, color = "light gray") + geom_point() +
    geom_line() + theme_bw() + scale_color_manual(values = c("#e41a1c", "#ff7f00",
    "#377eb8", "#4daf4a")) + ylab(expression(Local ~ instability ~ (CV[L]))) +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        legend.position = "none", legend.key = element_blank(), axis.text.x = element_text(angle = 45,
            vjust = 0.5)) + xlab("") + scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68",
    `2` = "1969-76", `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08"))

sync <- ggplot(all.c.split$synchrony, aes(year, (data), colour = cropid)) +
    geom_vline(xintercept = 2, lty = 2, color = "light gray") + geom_vline(xintercept = 3,
    lty = 2, color = "light gray") + geom_vline(xintercept = 6, lty = 2, color = "light gray") +
    geom_point() + geom_line() + theme_bw() + scale_color_manual(values = c("#e41a1c",
    "#ff7f00", "#377eb8", "#4daf4a")) + ylab(expression(Synchrony ~ (phi))) +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        legend.key = element_blank(), axis.text.x = element_text(angle = 45,
            vjust = 0.5), legend.key.size = unit(0.5, "cm"), legend.title = element_blank(),
        legend.position = c(0.8, 0.8)) + xlab("Time window") + scale_x_continuous(breaks = 1:6,
    labels = c(`1` = "1961-68", `2` = "1969-76", `3` = "1977-84", `4` = "1985-92",
        `5` = "1993-2000", `6` = "2001-08"))

global <- plot_grid(global, local, sync, align = c("v", "h"), ncol = 2)

global
```



Supplementary Figure 5: Stability of global crop production 1961-2008. Trends in global instability result from changes in both local instability, and synchronation in production trends. Dashed gray lines show time windows in which maximum negative deviations from mean production occured i.e. 1976 for soybean (-14%), 1983 for maize (-20%), 2002 for rice (-8%) and 2003 for wheat (-7%). Synchrony is a unitless metric running from completely synchronous local production trends (1) to completely assyncronous local production trends (0), and scales local to global instability through the following relationship: $\mathrm{CV}_G^2 = CV_L^2 \cdot \phi$.

4.5 Between crop synchrony

In this section we estimate the between crop synchrony, by utilizing the global time series of each crop,and converting these into crop calories, to identify if major crops have shown compensatory dynamics over the time series.

First we need to work out the kcal per tonnage of crop which we pull from Cassidy et al. [2]. Then we apply each crop to the framework developed above.

We also note that for this calorie analysis, we are interested in scaling the variance of the time series of global trends in calorie production for each crop, rather rather than representing production trends for individual production cells. We therefore adjust the axes labels to "Total calorie instability (CV_T)" and "Crop calorie instability (CV_C)" to reflect this difference, where, for completenes:

$CV_T = \sigma_T \div \mu_T$

and

$CV_C = \sigma_C \div \mu_T$

And where,

$\sigma_T = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j}}$

and

$\sigma_C = \sum_{i=1}^{n} \sqrt{x_{i,i}}$

In the above μ_T is the non-deterended mean of global calorie production, and n is the number of crops used in the analysis ($n = 4$).

To allow for easier interpretation of compensatory dynamics between crops we also plot the case of independent crop dynamics, utilizing the fact that when production trends are completely uncorrelated σ_T^2 is equal to:

$\sigma_I^2 = \sum_{i=1}^{n} x_{i,i}$

with

$\phi_{p=0} = \sigma_I^2 \div (\sum_{i=1}^{n} \sqrt{x_{i,i}})^2$

giving the synchrony metric for each time point, when $\rho = 0$.

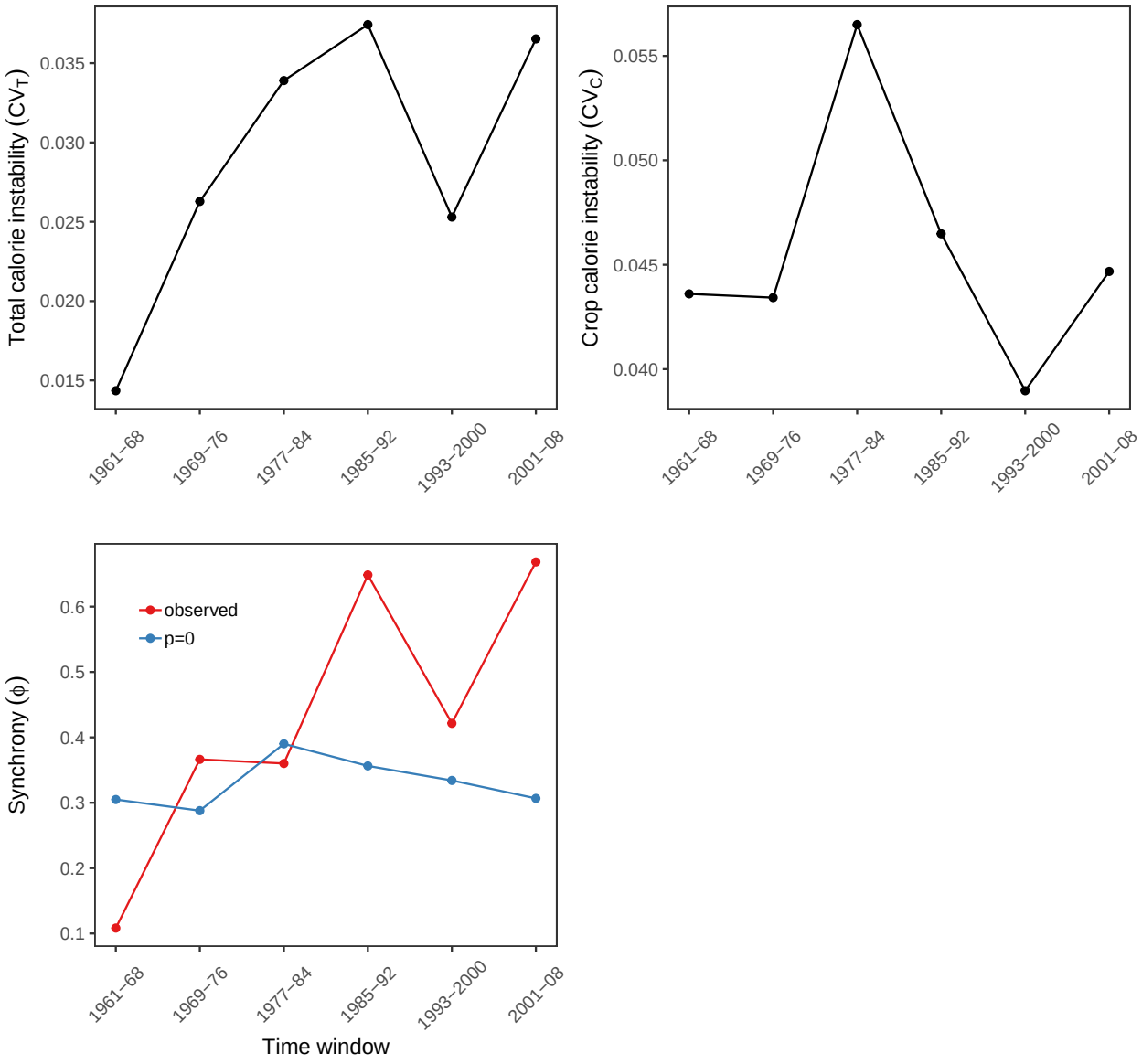
```
rm(list = ls())  #clear space
maize <- readRDS("maizeproduction100km2.rds")
rice <- readRDS("riceproduction100km2.rds")
soybean <- readRDS("soybeanproduction100km2.rds")
wheat <- readRDS("wheatproduction100km2.rds")

# compute the cal.sums
wheat.sum <- colSums(wheat, na.rm = T) * 3284
maize.sum <- colSums(maize, na.rm = T) * 3580.8026
soybean.sum <- colSums(soybean, na.rm = T) * 3596.49911
rice.sum <- colSums(rice, na.rm = T) * 2800

# all.crops
all.cals <- data.frame(rbind(wheat.sum, maize.sum, soybean.sum, rice.sum))
all.cals <- all.cals[, -49:-50]  #rmv co-ords

w.compl <- all.cals  #rename, as per above to keep the function the code the same below.


year <- 1961:2008  #set year

# split data
w.compl.dec <- list(w.compl[1:8], w.compl[9:16], w.compl[17:24], w.compl[25:32],
    w.compl[33:40], w.compl[41:48])

w.compl.dec.sum <- lapply(w.compl.dec, function(x) colSums(x))

detrend.w <- data.frame(t(apply(w.compl, 1, function(x) resid(loess(x ~ year)))))  #use loess for detrending

det.w.dec <- list(detrend.w[1:8], detrend.w[9:16], detrend.w[17:24], detrend.w[25:32],
    detrend.w[33:40], detrend.w[41:48])  #split data into windows

det.w.dec <- lapply(det.w.dec, t)  #transpose to get the grid cells on the cols

diff.cov <- lapply(lapply(det.w.dec, data.frame), cov)  #make covariance matrix of the grid cells.
diag.cov <- lapply(diff.cov, diag)  #get diagonals of covariance matrix = local variance

sqrt.diag.cov <- lapply(diag.cov, sqrt)  #sqrt of diags of covariance matrix = list of local sds

diff.var.glob <- unlist(lapply(diff.cov, sum))  #sum the covariance matrix. = global variance
diff.var.local <- unlist(lapply(diag.cov, sum))  #sum the diagonals = local variance

diff.sd.global <- sqrt(diff.var.glob)  #global sd
diff.sd.local <- unlist(lapply(sqrt.diag.cov, sum))  #local sd

diff.sync <- diff.var.glob/(diff.sd.local^2)  #synchrony

sync.p0 <- diff.var.local/(diff.sd.local^2)

mean.dec <- unlist(lapply(w.compl.dec.sum, mean))  #non-detrended global means

CVl <- diff.sd.local/mean.dec  #local instability
CVg <- diff.sd.global/mean.dec  #global instability

# bind all together in dataframe + save
vars <- c(rep("global variance", 6), rep("local variance", 6), rep("observed",
    6), rep("CVl", 6), rep("CVg", 6), rep("p=0", 6))
data <- c(diff.var.glob, diff.var.local, diff.sync, CVl, CVg, sync.p0)
year <- c(rep(1:6, 6))
all <- data.frame(vars, data, year)  #crop specific
```

4.5.1 Supplementary Figure 6

Finally we plot the between crop dynamics below. We in particular are interested in how global instability of calories has changed over the time series, and whether crops have compensated each other. We do not find much evidence that asynchrony between crops has stabilized the global variance in calorie production.

But we do find evidence that despite crop based calorie stability flat lining (e.g. 1961-1976), or even declining (1984-1992) aggregate calorie instability increased in both of these periods, and these increases were due to increased synchrony between crops.

```
library(ggplot2)  #load libraries
library(gridExtra)
library(cowplot)

all.c.split.cal <- split(all, all$vars)  #split buy variable to plot individsually
sync <- subset(all, vars %in% c("p=0", "observed"))

global.cal <- ggplot(all.c.split.cal$CVg, aes(year, (data))) + geom_point() +
    geom_line() + theme_bw() + ylab(expression(Total ~ calorie ~ instability ~
    (CV[T]))) + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    legend.position = "none", legend.key = element_blank(), legend.key.size = unit(0.5,
        "cm"), legend.title = element_blank(), axis.text.x = element_text(angle = 45,
        vjust = 0.5)) + xlab("") + scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68",
    `2` = "1969-76", `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08"))

local.cal <- ggplot(all.c.split.cal$CVl, aes(year, (data))) + geom_point() +
    geom_line() + theme_bw() + ylab(expression(Crop ~ calorie ~ instability ~
    (CV[C]))) + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    legend.position = "none", legend.key = element_blank(), axis.text.x = element_text(angle = 45,
        vjust = 0.5)) + xlab("") + scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68",
    `2` = "1969-76", `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08"))

sync.cal <- ggplot(sync, aes(year, (data), color = vars)) + geom_point() + geom_line() +
    theme_bw() + ylab(expression(Synchrony ~ (phi))) + scale_color_manual(values = c("#e41a1c",
    "#377eb8")) + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    legend.key = element_blank(), axis.text.x = element_text(angle = 45, vjust = 0.5),
    legend.key.size = unit(0.5, "cm"), legend.title = element_blank(), legend.position = c(0.2,
        0.8)) + xlab("Time window") + scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68",
    `2` = "1969-76", `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08"))

global.cal2 <- plot_grid(global.cal, local.cal, sync.cal, align = c("v", "h"),
    ncol = 2)
global.cal2
```



Supplementary Figure 6: Stability of global calorie production 1961-2008. Trends in global instability result from changes in both local instability, and synchronization in production trends.

4.6 Scenario planning

In this section we run an analysis to understand what the impact of synchronisation would be on the maximum deficits (percent negative deviations from the mean) observed for each of the major crops over the period of 1961-2008. This is a very simple approach that starts by converting residual production values into percentages of the mean trend. After this, we then compute the number of standard deviations that each of these deficits fall by computing the standard deviation of the residuals and dividing each residual value by that standard deviation. In the next step, the standard deviation of the time series of each of the crops under synchrony is determined. And then, to identify the residual under synchrony, the standard deviation of the synchronised time series are multiplied by the standard deviation distance that a deficit falls in the years of the maximum deficits. Finally these new residuals under synchrony are converted into percentages based on the non-detrended production time series.

4.6.1 Preliminary summaries

Here we identify the year with the maximum percent loss in production, the value of this deviation, and the number of standard deviations from the mean that this occurred. As we can see from the table the worst years occurred midway in the time series for maize (1983) and soybean (1976), and in the last decade of the time series for rice (2002) and wheat (2003). The observed largest percent losses from the mean trend are of -20% for maize, -14% for soybean, and -7% for both wheat, and -8% for rice. In total the worst years represented deviations of around -1.7σ for soybean, -2.9σ for maize, -3.6σ for rice and -2.2σ for wheat

```
rm(list = ls())
crops <- list("soybean", "maize", "rice", "wheat")
all.crop.final = data.frame()

for (i in 1:length(crops)) {
    crop <- crops[i]  #assign crop
    path = (paste(crop, "production100km2.rds", sep = ""))  #create path
    w <- readRDS(path)  #read in
    w <- subset(w, , -c(x, y))  #remove co-ordinates

    w.compl <- w[rowSums(is.na(w)) != ncol(w), ]  #remove complete cases of NA
    w.compl[is.na(w.compl)] <- 0  # replace NA with zeros
    w.compl.nd <- w.compl  #save a non-detrended version
    year <- 1961:2008

    # detrend using loess fits. doesn't matter if done local or global, but need
    # local for later.
    w.compl <- data.frame(t(apply(w.compl, 1, function(x) resid(loess(x ~ year)))))

    residual <- (colSums(w.compl))  #the residual on global scale
    actual.mean <- predict(loess(colSums(w.compl.nd) ~ year))  #the mean observed valuenon global scale
    perc.dev <- residual/actual.mean  #percent deviation
    sd.resid <- sd(residual)  #sd of the residuals
    sd.diff <- residual/sd.resid  #sd deviation

    ## call the values to output
    sddiff.max <- min(sd.diff)
    max.p.obs <- min(perc.dev)
    max.year <- rownames(data.frame(perc.dev))[which.min(apply(data.frame(perc.dev),
        MARGIN = 1, min))]  #year of the maximum deficit

    ## join into a dataframe
    vars <- c(rep("sddiff.max", 1), rep("max.p.obs", 1), rep("max.year", 1))
    data <- c(sddiff.max, max.p.obs, max.year)
    cropid <- c(rep(unlist(crop), 3))
    all <- data.frame(vars, data, cropid)  #crop specific
    all.crop.final <- rbind(all.crop.final, all)
}
```

```
all.crop.final

         vars                      data  cropid
1  sddiff.max         -1.78615527792799 soybean
2   max.p.obs         -0.13574644740093 soybean
3    max.year Soybean_1976_Area_ver8.nc soybean
4  sddiff.max         -2.95966606223038   maize
5   max.p.obs        -0.202261288910731   maize
6    max.year   Maize_1983_Area_ver8.nc   maize
7  sddiff.max         -3.62970643070119    rice
8   max.p.obs       -0.0800986244456992    rice
9    max.year    Rice_2002_Area_ver8.nc    rice
10 sddiff.max         -2.29320427307859   wheat
11  max.p.obs        -0.073532323687214   wheat
12   max.year   Wheat_2003_Area_ver8.nc   wheat
```

4.6.2 Mitigation strategies

Using the historical data, we constructed a scenario with complete synchronization of production trends for each of the four crops, and compared expected losses under this setting to the losses expected under the observed baseline trends. We set up our thought experiment to occur in the final year of the data set. To estimate the baseline losses, we used the number of standard deviations that the maximum losses fell over the 1961-2008 time series (-1.7σ for soybean, -2.9σ for maize, -3.6σ for rice and -2.2σ for wheat), to gain the lower bounds of a 100% historical prediction interval for productivity of this period. Then we estimated the losses under complete synchrony.

Given these two scenarios above, we work out the degree to which we would need to apply four mitigation solutions to bring synchronised losses back to the baseline. We split these mitigation strategies into variance reducing strategies and mean increasing strategies. Both of these strategies work to reduce the expected minima of global production, but they work in different ways. Variance reducing strategies work by making the system more stable overall, whereas mean increasing strategies, simply work to reduce the size of the deficit when expressed as the percent deviation from mean annual production trends.

Both types of strategies are grounded in potentially realistic interventions. For example, variance reducing strategies, can be implemented by diversifying genotypes, by adapting climate smart cropping systems, by using ecological engineering, or developing technological infrastructure to resist environmental stressors. We could focus variance reducing strategies worldwide in an "Increasing local stability" scenario, and only in breadbaskets using an "Increasing the stability of breadbaskets" scenario. Mean increasing strategies can be achieved through expansion of agricultural land, or more palatably with respect to biodiversity loss, through increasing yield ceilings and decreasing yield gaps. Like variance reducing interventions, mean increasing techniques are also largely spatially dependent. We focus mean increasing strategies on a "Closing production gaps" scenario, and on a breadbaskets using an "Raising production ceilings" scenario. The details of the set- up and four strategies run in this thought experiment are shown below.

1. The baseline loss (B) is defined as $\frac{\sigma_G \times l}{\hat{y}_n}$, where $\hat{y}_n$ is equal to to the mean production in the final year of the data set, estimated using $f(t_i)$ from a loess regression: $y_i = f(t_i) + e_i$, where y_i is the global production for year i, t_i is the year, $f()$ is a smooth function, and e is the residual error. And where $l = \frac{\min e}{\sigma_G}$, indicating the number of standard deviations that the minimum residual fell over the production time series. The global standard deviation, σ_G, is equal to $\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j}}$. This special case of baseline losses can be simplified further to $\frac{\min e}{\hat{y}_n}$.

2. Loss under complete synchrony (S) are similarly defined as $\frac{\sigma_L \times l}{\hat{y}_n}$, where σ_L is equal to $\sum_{i=1}^{n} \sqrt{x_{i,i}}$. The numerator of this equation, multiplies the maximum loss in units of σ_G, by the global standard deviation of production under synchrony, to obtain a maximum negative deviation from the mean under the complete synchronisation of local production trends.

3. The factor increase in local stability across all producing locations worldwide that would be needed to bring production losses back to baseline loss is then equal to $\left(\frac{\sigma_L}{(B \times \hat{y}_n) \div l}\right)^2$.

4. The factor increase in local stability across breadbaskets that would be needed to bring production losses back to baseline loss is then equal to $\left(\frac{\sigma_{L>c}}{(B_2 \times \hat{y}_n) \div l}\right)^2$, where $B_2 = B - S_2$, and where $S_2 = \frac{\sigma_{Lc<c} \times l}{\hat{y}_n}$. In these formulas, c indicates the percentile of total local production around which the production cells in the data is partitioned to compute σ_L. For this thought experiment we split the data around the 80th percentile - to work out the factor reduction in variance within highly productive areas needed to counter the complete increase in synchrony worldwide. For a given value of c only under conditions where $B > S_2$ will actions in these breadbaskets able to completely mitigate losses due to synchrony.

5. The factor increase in raising production ceilings is then equal to $\frac{((\sigma_L \times l) \div B) - \hat{y}_{n>c}}{\hat{y}_{n<c}}$, where $\hat{y}_{n>c}$ represents the global mean production in the final year of the data for the subset of the data partitioned around c. For this scenario we set where c equal to 80. In other words, asking the question in how much of an increase in local production for the most productive locations on the planet would be needed to offset the losses.

6. The factor increase in closing production gaps is equal to $\frac{((\sigma_L \times l) \div B) - \hat{y}_{n<c}}{\hat{y}_{n>c}}$. For this scenario we set where c equal to 50, to identify the question of how much of an increase in local production for the least productive locations on the planet would be needed to offset the losses.

The code below computes each of these scenarios for each crop in turn, and prints the results in a table. As we can see below most effective strategies (i.e. those that require the least change from current settings), mean increasing strategies only seem to work well when applied in the most productive locations (and less well when applied to least productive locations), and variance reducing strategies only seem to work well when distributed across the whole planet (and less well when applied only to breadbaskets).

```r
rm(list = ls())  #clear space
crops <- list("maize", "rice", "soybean", "wheat")
all.crop.scen.te2 = data.frame()
for (i in 1:length(crops)) {
    crop <- crops[i]  #assign crop
    path = (paste(crop, "production100km2.rds", sep = ""))  #create path
    w <- readRDS(path)  #read in
    w <- subset(w, , -c(x, y))  #remove co-ordinates
    w.compl <- w[rowSums(is.na(w)) != ncol(w), ]  #remove complete cases of NA
    w.compl[is.na(w.compl)] <- 0  # replace NA with zeros
    w.compl.nd <- w.compl  #save a non-detrended version
    year <- 1961:2008
    w.compl <- data.frame(t(apply(w.compl, 1, function(x) resid(loess(x ~ year)))))  #detrend the data
    residual <- (colSums(w.compl))  #the residual on global scale
    actual.mean <- predict(loess(colSums(w.compl.nd) ~ year))  #the mean observed value on global scale
    sd.resid <- sd(residual)  #sd of the residuals
    sd.diff.min <- min(residual/sd.resid)  #sd deviation

    # 1. Baseline losses
    perc.dev <- (sd.resid * sd.diff.min)/actual.mean  #percent deviation under baseline

    # 2. Losses under synchrony
    sdlocal.p1 <- sum(apply(w.compl, 1, sd, na.rm = T))  #the sd under complete synchrony
    sync.resid <- sd.diff.min * sdlocal.p1  #the residuals under synchrony
    perc.dev.sync <- sync.resid/actual.mean  #get the deviations under synchrony, i.e. WCS

    # 3. Global variance reduction
    Fact.variance.red <- (sum(sqrt((apply(w.compl, 1, sd, na.rm = T)^2)))/((perc.dev[48] *
        actual.mean[48])/sd.diff.min))^2

    # 4. Breadbasket variance reduction
    vars <- as.vector(apply(w.compl, 1, var, na.rm = T))  #variance in production
    local.sum <- as.vector(rowSums(w.compl.nd, na.rm = FALSE, dims = 1))  #sum of production
    quantile80 <- ifelse(local.sum >= quantile(local.sum, probs = 0.8), 1, 0)  #id top producers
    sep.var <- split(vars, quantile80)  #split the vars by producer quantiles
    y <- ((sum(sqrt(sep.var[["0"]])) * sd.diff.min)/actual.mean)[48]  #loss due to low producers variance
    b2 <- perc.dev[48] - y  #the top producers target loss
    Fact.variance.red.bb <- ((sum(sqrt(sep.var[["1"]])))/((b2 * actual.mean[48])/sd.diff.min))^2

    # 5. Raise production ceilings
    vars <- as.vector(apply(w.compl, 1, var, na.rm = T))  #variance in production
    local.sum <- as.vector(rowSums(w.compl.nd, na.rm = FALSE, dims = 1))  #sum of production
    quantile80 <- ifelse(local.sum >= quantile(local.sum, probs = 0.8), 1, 0)  #id top producers
    split.pro <- split(w.compl.nd, quantile80)
    p1 <- predict(loess(colSums(split.pro[["0"]]) ~ year))[48]
    p2 <- predict(loess(colSums(split.pro[["1"]]) ~ year))[48]
    Fact.mean.inc.bb <- ((sync.resid/perc.dev[48]) - p1)/p2

    # 6. Close production gaps
    quantile50 <- ifelse(local.sum >= quantile(local.sum, probs = 0.5), 0, 1)  #id top/bottom producers
    split.pro.50 <- split(w.compl.nd, quantile50)
    p1 <- predict(loess(colSums(split.pro.50[["0"]]) ~ year))[48]
    p2 <- predict(loess(colSums(split.pro.50[["1"]]) ~ year))[48]
    Fact.mean.inc.low <- ((sync.resid/perc.dev[48]) - p1)/p2

    # join into a dataframe
    vars <- c("Baseline.loss", "Sync.loss", "Global variance reduction", "Bread-basket variance reduction",
        "Raising production ceiling", "Closing production gaps")
    data <- c(perc.dev[48], perc.dev.sync[48], Fact.variance.red, Fact.variance.red.bb,
        Fact.mean.inc.bb, Fact.mean.inc.low)
    cropid <- c(rep(unlist(crop), 6))
    all <- data.frame(Scenario = vars, Loss_or_Factor.change = data, Crop = cropid)  #crop specific
    all.crop.scen.te2 <- rbind(all.crop.scen.te2, all)
}
```

```
all.crop.scen.te2  #print results
```

	Scenario	Loss_or_Factor.change	Crop
1	Baseline.loss	-0.106	maize
2	Sync.loss	-0.342	maize
3	Global variance reduction	10.353	maize
4	Bread-basket variance reduction	36.782	maize
5	Raising production ceiling	3.565	maize
6	Closing production gaps	136.788	maize
7	Baseline.loss	-0.074	rice
8	Sync.loss	-0.252	rice
9	Global variance reduction	11.678	rice
10	Bread-basket variance reduction	96.411	rice
11	Raising production ceiling	3.801	rice
12	Closing production gaps	199.187	rice
13	Baseline.loss	-0.041	soybean
14	Sync.loss	-0.174	soybean
15	Global variance reduction	18.189	soybean
16	Bread-basket variance reduction	398.680	soybean
17	Raising production ceiling	4.743	soybean
18	Closing production gaps	204.504	soybean
19	Baseline.loss	-0.071	wheat
20	Sync.loss	-0.326	wheat
21	Global variance reduction	21.136	wheat
22	Bread-basket variance reduction	63.855	wheat
23	Raising production ceiling	5.620	wheat
24	Closing production gaps	158.572	wheat

4.7 Climate analysis

4.7.1 Climate synchrony

In this section we estimate the synchrony for each climate variable for each crop. We adapt the loop outlined above for the crop production case but for both precipitation and temperature. We note that the temperature is represented as means not cumulative sums, but this will suffice for calculation of the synchrony metric. We run this for the complete crop calender case.

```
rm(list = ls())  #clear space

files <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/mask_Del_100km_eck4/hybrid",
    full.names = T)  #write the list.
files.short <- list.files("/Users/pumpkinjr/Sync/academic/projects/UBCload/Data/masks2/mask_Del_100km_eck4/hybrid")

all.crop.clim = data.frame()
for (i in 1:length(files.short)) {
    file <- files.short[i]  #grab name
    w <- readRDS(files[i])  #read in file
    w <- subset(w, , -c(x, y))  #remove co-ordinates

    w.compl <- w[rowSums(is.na(w)) != ncol(w), ]  #remove complete cases of NA
    w.compl[is.na(w.compl)] <- 0  #replace NA with zeros
    year <- 1961:2008
    detrend.w <- data.frame(t(apply(w.compl, 1, function(x) resid(loess(x ~
        year)))))  #use loess for detrending
    det.w.dec <- list(detrend.w[1:8], detrend.w[9:16], detrend.w[17:24], detrend.w[25:32],
        detrend.w[33:40], detrend.w[41:48])  #split data into windows

    det.w.dec <- lapply(det.w.dec, t)  #transpose to get the grid cells on the cols
    diff.cov <- lapply(lapply(det.w.dec, data.frame), cov)  #make covariance matrix of the grid cells.
    diag.cov <- lapply(diff.cov, diag)  #get diagonals of covariance matrix = local variance
    sqrt.diag.cov <- lapply(diag.cov, sqrt)  #sqrt of diags of covariance matrix = list of local sds
    diff.var.glob <- unlist(lapply(diff.cov, sum))  #sum the covariance matrix. = global variance
    diff.var.local <- unlist(lapply(diag.cov, sum))  #sum the diagonals = local variance
    diff.sd.global <- sqrt(diff.var.glob)  #global sd
    diff.sd.local <- unlist(lapply(sqrt.diag.cov, sum))  #local sd
    diff.sync <- diff.var.glob/(diff.sd.local^2)  #synchrony
    # bind all together in dataframe + save -- this should produce a dataframe
    # identifying the synchrony in ppt and temp for the growing season and
    # locations of the crops identified in the above analysis.
    vars <- c(rep("synchrony", 6))
    data <- c(diff.sync)
    year <- c(1:6)
    cropid <- c(rep(unlist(file), 6))
    all <- data.frame(vars, data, year, cropid)  #crop specific
    all.crop.clim <- rbind(all.crop.clim, all)
}

## then save
saveRDS(all.crop.clim, "sync.climate.rds")
```

4.7.2 Graphical analysis

In this section we will perform a graphical analysis to assess the temporal patterns in climate synchrony. We will also assess if there is an indication that the patterns in production synchrony match those in climate synchrony.

First we need to join all of the data into a common data frame for plotting.

```
all.crop <- readRDS("allcrop.rds")

sync <- subset(all.crop, vars == "synchrony")
all.clim <- readRDS("sync.climate.rds")

all.clim$var <- ifelse(grepl("ppt", all.clim$cropid) == T, "Precipitation",
    "Temperature")
all.clim$crop <- ifelse(grepl("maize", all.clim$cropid) == T, "maize", ifelse(grepl("soy",
    all.clim$cropid) == T, "soybean", ifelse(grepl("wheat", all.clim$cropid) ==
    T, "wheat", "rice")))
all.clim$prod <- rep(sync$data, 2)
```

Then we plot the time-series of synchrony in meteorology for each crop's growing location. As can be seen below, synchrony in climate variables – such as cumulative growing season precipitation and average growing season temperature – in general do not match the patterns in production synchrony. The temperature trends look potentially oscillatory. The only climate trend which looks like it might follow the same dynamics as crop production, is precipitation and rice production.

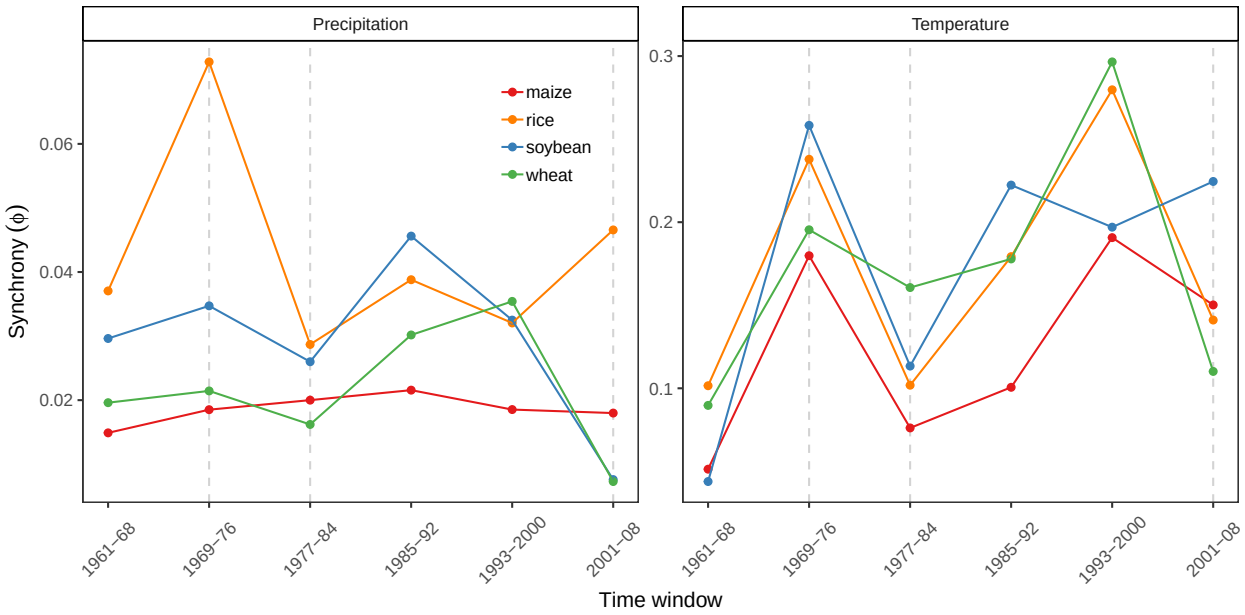
```
clim.sync <- ggplot(all.clim, aes(year, (data), colour = crop)) + geom_vline(xintercept = 2,
    lty = 2, color = "light gray") + geom_vline(xintercept = 3, lty = 2, color = "light gray") +
    geom_vline(xintercept = 6, lty = 2, color = "light gray") + geom_point() +
    geom_line() + theme_bw() + scale_color_manual(values = c("#e41a1c", "#ff7f00",
    "#377eb8", "#4daf4a")) + ylab(expression(Synchrony ~ (phi))) + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), legend.key = element_blank(), axis.text.x = element_text(angle = 45,
        vjust = 0.5), legend.key.size = unit(0.5, "cm"), legend.title = element_blank(),
    legend.position = c(0.4, 0.8), strip.background = element_rect(colour = "black",
        fill = "white")) + xlab("Time window") + scale_x_continuous(breaks = 1:6,
    labels = c(`1` = "1961-68", `2` = "1969-76", `3` = "1977-84", `4` = "1985-92",
        `5` = "1993-2000", `6` = "2001-08")) + facet_wrap(~var, scales = "free")
```

Below we overlay the production and climate synchrony trends. Indeed as we can see from these plots, rice and precipitation do seem to show consistent signs of inflection in dynamics of synchrony. However, we note some departure from climate synchrony and production synchrony for rice in the most recent year – where production synchrony rises much more sharply that climate synchrony.

```
clim.sync.l <- reshape2::melt(all.clim, id.vars = c("year", "cropid", "crop",
    "var"), measure.vars = c("data", "prod"))

clim.sync.l$source <- ifelse(clim.sync.l$variable == "data", "Climate", "Production")

clim.sync.over <- ggplot(clim.sync.l, aes(year, (value), colour = source)) +
    geom_point() + geom_line() + theme_bw() + scale_color_manual(values = c("#e41a1c",
    "#377eb8")) + ylab(expression(Synchrony ~ (phi))) + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), legend.key = element_blank(), axis.text.x = element_text(angle = 45,
        vjust = 0.5), legend.key.size = unit(0.5, "cm"), legend.title = element_blank(),
    strip.background = element_rect(colour = "black", fill = "white")) + xlab("Time window") +
    scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68", `2` = "1969-76",
        `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08")) +
    facet_wrap(var ~ crop, scales = "free", ncol = 4)
```

4.7.3 Supplementary Figure 7

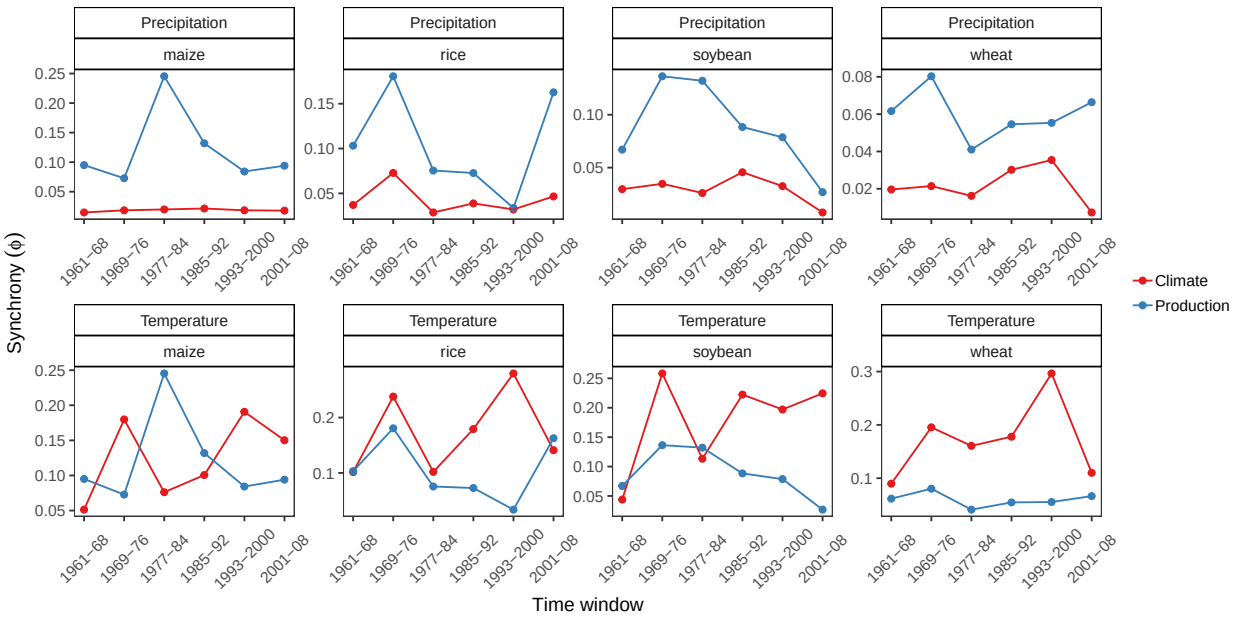
```
clim.sync
```



Supplementary Figure 7: Synchrony in growing season climate for major crops between 1961-2008

4.7.4 Supplementary Figure 8

```
clim.sync.over
```



Supplementary Figure 8: Trends in synchrony in growing season climate and production for major crops across the world between 1961-2008

4.7.5 Correlation analysis

Here we run some a basic correlation analysis for the climate versus production synchrony metrics. We note caution on over interpreting this inference due to the near decadal window size, and short (50 year) time series. This renders any inference massively under powered. Indeed, for a sample size of six, the probability of failing to reject the null when it is false (a type II error) when the population statistic is even as high as 0.8, at an α of 0.05, is as high as 40-50% [3]. Indeed, the corresponding approximate estimate of power under this scenario is shown below.

```
pwr::pwr.r.test(n = 6, r = 0.8, sig.level = 0.05)
```

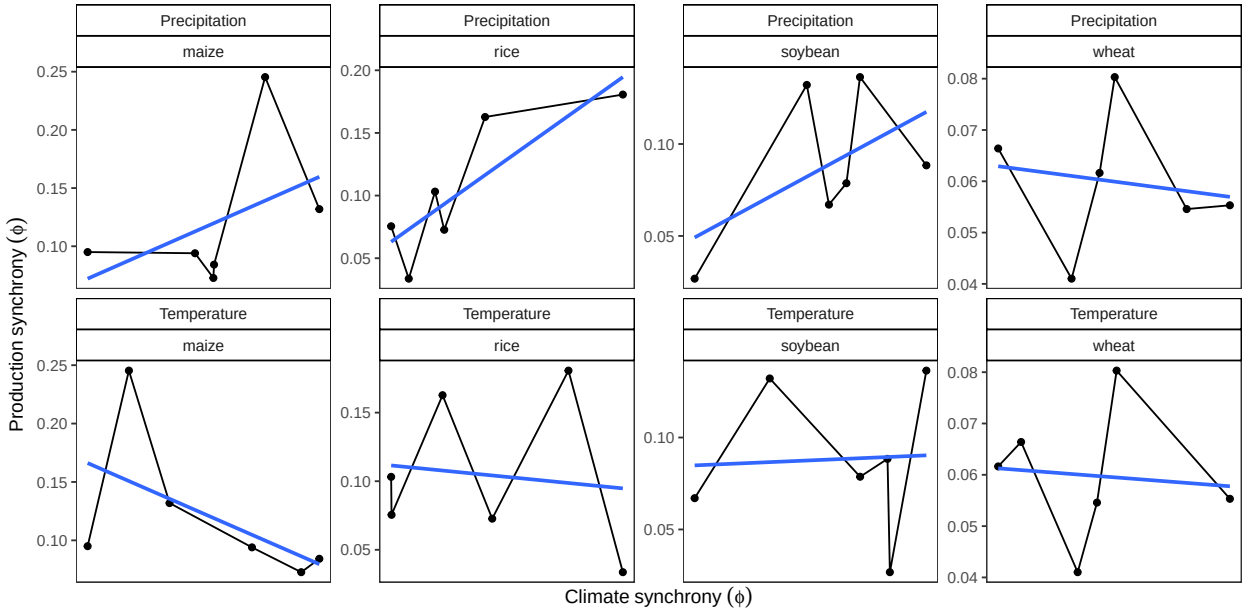
```
     approximate correlation power calculation (arctangh transformation)

              n = 6
              r = 0.8
      sig.level = 0.05
          power = 0.53
    alternative = two.sided
```

Below plot the scatter plots of the data.

```
clim.sync.scat <- ggplot(all.clim, aes(data, (prod))) + geom_point() + geom_line() +
    geom_smooth(method = "lm", se = F) + theme_bw() + # scale_color_manual(values = c('#e41a1c','#377eb8'))+
ylab(expression(Production ~ synchrony ~ (phi))) + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), legend.key = element_blank(), axis.text.x = element_text(angle = 45,
        vjust = 0.5), legend.key.size = unit(0.5, "cm"), legend.title = element_blank(),
    strip.background = element_rect(colour = "black", fill = "white")) + xlab(expression(Climate ~
    synchrony ~ (phi))) + scale_x_continuous(breaks = 1:6, labels = c(`1` = "1961-68",
    `2` = "1969-76", `3` = "1977-84", `4` = "1985-92", `5` = "1993-2000", `6` = "2001-08")) +
    facet_wrap(var ~ crop, scales = "free", ncol = 4)
```

4.7.6 Supplementary Figure 9

```
clim.sync.scat
```



Supplementary Figure 9: Scatter plots of synchrony in growing season climate and production for major crops across the world between 1961-2008

Then we estimate both Pearson's r and Spearman's rho, which identify linear and monotonic relationships between production and climate synchrony for each crop, respectively, and then estimate the 95% intervals for these. First we estimate Pearson's correlation below. As expected we can see the intervals around all the estimates are large, and for the case of rice and precipitation this is also true ranging between 0.09 - 0.98.

```
library(plyr)
func.pearson <- function(xx) {
    COR = cor.test(xx$data, xx$prod, method = "pearson")
    return(data.frame(COR$estimate, COR$p.value, COR$conf.int[1], COR$conf.int[2]))
}
out.p <- ddply(all.clim, .(cropid), func.pearson)
out.p
```

```
                            cropid COR.estimate COR.p.value
1     pptmaizeHy.mask.rds100km2.rds        0.455       0.364
2      pptriceHy.mask.rds100km2.rds        0.841       0.036
3  pptsoybeansHy.mask.rds100km2.rds        0.545       0.264
4     pptwheatHy.mask.rds100km2.rds       -0.162       0.760
5    tempmaizeHy.mask.rds100km2.rds       -0.553       0.255
6     tempriceHy.mask.rds100km2.rds       -0.121       0.819
7 tempsoybeansHy.mask.rds100km2.rds        0.050       0.925
8    tempwheatHy.mask.rds100km2.rds       -0.094       0.859
  COR.conf.int.1. COR.conf.int.2.
1          -0.565            0.93
2           0.093            0.98
3          -0.478            0.94
4          -0.860            0.75
5          -0.942            0.47
6          -0.849            0.77
7          -0.794            0.83
8          -0.841            0.78
```

We also estimate the correlation for Spearmnan's rho too.

```
func.spearman <- function(xx) {
    COR = cor.test(xx$data, xx$prod, method = "spearman")
    return(data.frame(COR$estimate, COR$p.value))
}
out.s <- ddply(all.clim, .(cropid), func.spearman)
```

Confidence intervals around Spearman's rho using the Fisher z transform with the population variance adjustment from Bonett and Wright [1] are given below.

```
out.s$z.low<-atanh(out.s$COR.estimate) -  #z transform
  qnorm(0.975)*(sqrt((1+out.s$COR.estimate^2/2)/(6-3))) #lower
out.s$z.up<-atanh(out.s$COR.estimate)+
  qnorm(0.975)*(sqrt((1+out.s$COR.estimate^2/2)/(6-3)))
out.s$r.lower<-tanh(out.s$z.low)
out.s$r.upper<-tanh(out.s$z.up)
out.s
```

```
                            cropid COR.estimate COR.p.value z.low z.up
1     pptmaizeHy.mask.rds100km2.rds        0.429        0.42 -0.72 1.64
2      pptriceHy.mask.rds100km2.rds        0.714        0.14 -0.37 2.16
3  pptsoybeansHy.mask.rds100km2.rds        0.543        0.30 -0.60 1.82
4     pptwheatHy.mask.rds100km2.rds       -0.143        0.80 -1.28 0.99
5    tempmaizeHy.mask.rds100km2.rds       -0.771        0.10 -2.31 0.27
6     tempriceHy.mask.rds100km2.rds       -0.257        0.66 -1.41 0.89
7 tempsoybeansHy.mask.rds100km2.rds        0.257        0.66 -0.89 1.41
8    tempwheatHy.mask.rds100km2.rds       -0.029        1.00 -1.16 1.10
  r.lower r.upper
1   -0.62    0.93
2   -0.36    0.97
3   -0.54    0.95
4   -0.86    0.76
5   -0.98    0.26
6   -0.89    0.71
7   -0.71    0.89
8   -0.82    0.80
```

5 Session information

```
sessionInfo()
```

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS  10.14

Matrix products: default
BLAS: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBLAS.dylib
LAPACK: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libLAPACK.dylib

locale:
[1] en_CA.UTF-8/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] plyr_1.8.4         gridExtra_2.3      rgdal_1.3-3
 [4] cowplot_0.9.3      ggplot2_3.0.0      tidyr_0.8.1
 [7] maptools_0.9-3     raster_2.6-7       sp_1.3-1
[10] RColorBrewer_1.1-2 knitr_1.20         RevoUtils_11.0.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.18     pillar_1.3.0     compiler_3.5.1   formatR_1.5
 [5] highr_0.7        bindr_0.1.1      tools_3.5.1      digest_0.6.15
 [9] evaluate_0.11    tibble_1.4.2     gtable_0.2.0     lattice_0.20-35
[13] pkgconfig_2.0.1  rlang_0.2.1      bindrcpp_0.2.2   pwr_1.2-2
[17] withr_2.1.2      stringr_1.3.1    dplyr_0.7.6      tidyselect_0.2.4
[21] glue_1.3.0       R6_2.2.2         foreign_0.8-70   reshape2_1.4.3
[25] purrr_0.2.5      magrittr_1.5     scales_0.5.0     assertthat_0.2.0
[29] colorspace_1.3-2 labeling_0.3     stringi_1.2.4    lazyeval_0.2.1
[33] munsell_0.5.0    crayon_1.3.4
```

References

[1] DG Bonnet and TA Wright. "Sample size requirements for estimating pearson, kendall and spearman correlations". In: *Psychometrika* 65.1 (2000), pp. 23–28.

[2] Emily S Cassidy et al. "Redefining agricultural yields: from tonnes to people nourished per hectare". In: *Environmental Research Letters* 8 (2013), p. 034015. ISSN: 1748-9326. DOI: 10.1088/1748-9326/8/3/034015. URL: http://iopscience.iop.org/1748-9326/8/3/034015/article/.

[3] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 1988.

[4] Microsoft Corporation. *checkpoint: Install Packages from Snapshots on the Checkpoint Server for Reproducibility*. R package version 0.4.4. 2018. URL: https://github.com/RevolutionAnalytics/checkpoint.

[5] Chad Monfreda, Navin Ramankutty, and Jonathan A. Foley. "Farming the planet: 2. Geographic distribution of crop areas, yields, physiological types, and net primary production in the year 2000". In: *Global Biogeochemical Cycles* 22.1 (). DOI: 10.1029/2007GB002947. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2007GB002947. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2007GB002947.

[6] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2015. URL: https://www.R-project.org/.

[7] Deepak K. Ray et al. "Climate variation explains a third of global crop yield variability". In: *Nature Communications* 6 (2015), p. 5989. ISSN: 2041-1723. DOI: 10.1038/ncomms6989. URL: http://www.nature.com/doifinder/10.1038/ncomms6989.

[8] Deepak K Ray et al. "Recent patterns of crop yield growth and stagnation." In: *Nature communications* 3 (2012), p. 1293. ISSN: 2041-1723. DOI: 10.1038/ncomms2296. URL: http://www.ncbi.nlm.nih.gov/pubmed/23250423.

[9] William J. Sacks et al. "Crop planting dates: an analysis of global patterns". In: *Global Ecology and Biogeography* 19.5 (), pp. 607–620. DOI: 10.1111/j.1466-8238.2010.00551.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1466-8238.2010.00551.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1466-8238.2010.00551.x.

[10] Shaopeng Wang and Michel Loreau. "Ecosystem stability in space: alpha, beta and gamma variability". In: *Ecology Letters* 17.8 (2014), pp. 891–901. ISSN: 14610248. DOI: 10.1111/ele.12292.

[11] C. J. Willmott and K Matsuura. *Terrestrial Air Temperature and Precipitation: Monthly and Annual Time Series (1950 - 1999)*. 2001. URL: http://climate.geog.udel.edu/~climate/html_pages/README.ghcn_ts2.html (visited on 11/01/2018).

[12] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.20. 2018. URL: https://CRAN.R-project.org/package=knitr.