Complete Code Book: c1-12

Contents

1 Fetching land cover data

Fetching regional and global landcover datasets to be analyzed relative to the SA reference dataset: the MODIS 2011 landcover product for South Africa was downloaded, the GLC-SHARE dataset, the 2009 GlobCover product, and South Africa's 2009 landcover product.

1.1 Download landcover datasets

```r
library(lmisc)  # custom package on GitHub with helper functions
library(RCurl)
library(httr)
```

```r
rm(list = ls())
p_root <- proj_root("croplandbias")
p_dl <- fp(p_root, "croplandbias/external/ext_data/landcover/")
p_orig <- fp(p_root, "croplandbias/external/ext_data/landcover/original")
```

1.1.1 GLC-Share

Starting with the improved/fusion dataset, GLC-Share. Note: the original version of analyses was looking at the geo-wiki dataset, but a coding error referred geowiki to GLC-Share, which was also initially tested, thus error assessments and downstream tests where performed on GLC-Share, not geo-wiki. Both are hybrid-fusion products, therefore serve the same role in analysis.

```r
setwd(p_dl)
path <- fp(p_dl, "glc_share")
url <- paste0("http://www.fao.org/geonetwork/srv/en/resources.get?id=47948&",
              "fname=GlcShare_v10_02.zip&access=private")
download.file(url, method = "auto", destfile = "glc_share.zip")
unzip("glc_share.zip", exdir = "glc_share")
```

1.1.2 Globcover 2009

```r
path <- fp(p_dl, "globcover_2009")
url <- "http://due.esrin.esa.int/files/Globcover2009_V2.3_Global_.zip"
download.file(url, method="auto", destfile = "globcover_2009.zip")
unzip("globcover_2009.zip", exdir = "globcover_2009")
```

1.1.3 MODIS landcover

```r
path <- fp(p_dl, "MCD12Q1/")
dir.create(fp(p_dl, "MCD12Q1/"))

# set up tiles we need and list of corresponding downloads
tiles <- expand.grid("h" = c(19, 20), "v" = c(11, 12))
tiles <- apply(tiles, 1, function(x) paste("h", x[1], "v", x[2], sep = ""))
url <- "https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.051/2011.01.01/"
tile.names <- getURL(url, verbose=TRUE, dirlistonly = TRUE)
tile.names1 <- strsplit(tile.names, "alt")[[1]]
tile.names2 <- unique(substr(tile.names1, 20, 64))
modnames <- tile.names2[sapply(tiles, function(x) grep(x, tile.names2))]

# download them
lapply(modnames, function(x) {  # x <- modnames[1]
  url2 <- paste0(url, x)
  GET(url2,
    authenticate("lestes", "hkPwMHdsKq8DxfrG"),
    write_disk(fp(path, x)))
})
```

1.1.4 South Africa's landcover database (30 m)

```
path <- fp(p_dl, "sa_lc_2009/")
url <- "http://planet.uwc.ac.za/BGISdownloads/landcover_2009.zip"
download.file(url, method ="auto", destfile = "landcover_2009.zip")  # painful
unzip("landcover_2009.zip", exdir = "sa_lc_2009")
```

2 Pre-processing reference cropland data layer

The following describes the initial pre-processing of landcover datasets that was undertaken for an analysis of error in landcover data.

Much of the original analysis was undertaken on the Mapping Africa server, using postgis/postgres as well as pprepair to do much of the cleaning and conversion of the GTI (reference) dataset into a 1 km resolution cropland percentage cover dataset. The following steps were taken:

2.1 Cleaning up fields in GTI data

This was done for both the 2007 and 2011 datasets, using a script that pulled out the fields, found the bad ones, read them into an R data object, ran pprepair over it, and then stuck the cleaned geometries back into a postgis table holding the full dataset. The intersection was then performed.

```
source("grid-fields-intersect-postgis.R")  # run on mapper.princeton.edu
```

2.2 Calculating field areas

The next step was running the following script to calculate the area for different cropcover classes in each 1 km^2 pixel. This created two core gridded datasets, cover2007.tif and cover2011.tif.

```
source("grid-fields-intersect-analyze.R")  # run on mapper.princeton.edu
```

Note: both scripts are available as system files under the sub-folder, and can be found using the following commands.

```
library(croplandbias)
system.file("extR", "grid-fields-intersect-postgis.R",
            package = "croplandbias")
system.file("extR", "grid-fields-intersect-analyze.R",
            package = "croplandbias")
```

3 Pre-processing landcover

Pre-processing global landcover datasets to be analyzed relative to the SA reference dataset: the MODIS 2011 landcover product for South Africa was downloaded, the GLC-SHARE dataset, the 2009 GlobCover product, and South Africa's 2009 landcover product. These were converted to 1 km^2 percentage cropland estimates after making several versions of MODIS and GlobCover products to account for mixed cropland classes. A further processing was applied to the two GTI derived cover images.

Additonal note: This part of the analysis was undertaken on the Linux VM mapper.princeton.edu, using `gdal 1.11`. For full reproducibility of results, this version should be used (`gdalwarp` appears to have different behaviour in older versions when no `-te` (extent) values are passed, which makes the warp slightly different between old and new versions due to different extents after warping).

3.1 Analyses

3.1.1 Set-up

```r
library(croplandbias)
library(gdalUtils)
library(RCurl)
library(httr)
# library(RPostgreSQL) # used for initial extraction of sagrid
library(rgeos)

# Paths
p_root <- proj_root("croplandbias")
p_data <- fp(p_root, "croplandbias/inst/extdata/landcover")
p_dl <- fp(p_root, "croplandbias/external/ext_data/landcover/")
p_orig <- fp(p_root, "croplandbias/external/ext_data/landcover/original")

# Note: installed from homebrew gdal 1.11.5, which is older than version with
# QGIS on Sierra (2.1)
gdal_setInstallation(search_path = "/usr/local/bin", rescan = TRUE)
getOption("gdalUtils_gdalPath")[[1]]$path
```

3.1.2 Background data, projections, database connections, and SA grid

```r
# Get SA shape
data(sashp)
sa_buf <- gBuffer(sashp, width = 3000)

# projections, etc
prjstr <- projection(sashp)
epsg <- data.table(make_EPSG())
gcsstr <- epsg[like(code, "4326"), prj4]

# Convert SA to GCS for cropping extent
sa_buf_gcs <- spTransform(sa_buf, CRSobj = CRS(gcsstr))

# Connection (deprecated, from old version of mapper.princeton.edu, since
# updated to have whole Africa grid)
# drv <- dbDriver("PostgreSQL")
# con <- dbConnect(drv, dbname = "SouthAfricaSandbox", user = "postgis",
#                  password = "P0stG1S")

# Create 1 km grid from sa1kilo to rectify grids against that
# sql <- paste("SELECT gid, id, ST_AsText(ST_Centroid(geom)) as center",
#              "FROM sa1kilo")
# geom.tab <- dbGetQuery(con, sql)
# coord.mat <- do.call(rbind, lapply(1:nrow(geom.tab), function(y) {
#   strip <- gsub("POINT|\\(|\\)", "", geom.tab[y, 3])
#   coord.mat <- do.call(rbind, strsplit(strsplit(strip, ",")[[1]], " "))
# }))
# class(coord.mat) <- "numeric"
# point.tab <- cbind(geom.tab[, 1:2], coord.mat)
# colnames(point.tab)[3:4] <- c("x", "y")
# pointsXYZ <- point.tab[, c(3:4, 2)]
# r <- rasterFromXYZ(pointsXYZ)
# projection(r) <- sa_buf@proj4string
# sa_r <- writeRaster(r, filename = fp(p_data, "sagrid.tif"),
#                     overwrite = TRUE)
sa_r <- raster(fp(p_data, "sagrid.tif"))
```

3.1.3 Bring in landcover datasets

Let's start with the improved/fusion dataset, GLC-Share. Note: the original version of analyses was looking at the geo-wiki dataset, but a coding error referred geowiki to GLC-Share, which was also initially tested, thus error assessments and downstream tests where performed on GLC-Share, not geo-wiki. Both are hybrid-fusion products, therefore serve the same role in analysis.

```r
setwd(p_dl)
path <- fp(p_dl, "glc_share")
r <- raster(fp(path, "glc_shv10_02.Tif"))
crs(r) <- CRS(gcsstr)
r <- crop(r, y=sa_buf_gcs, file = fp(path, "glcSA.tif"),
          overwrite = TRUE)  # crop to SA
gdalwarp(srcfile = filename(r), overwrite = TRUE,
         dstfile = fp(path, "glcSA_alb.tif"), t_srs = prjstr,
         tr = c(1000, 1000), r = "bilinear")
r <- raster(fp(path, "glcSA_alb.tif"))

# Warp to SA grid
r <- resample(r, sa_r, method = "bilinear", overwrite = TRUE,
              filename = fp(path, "glcSA_alb_rect.tif"))
r <- mask(r, sa_r, file = fp(p_data, "glcsa_masked.tif"),
          overwrite = TRUE)
```

3.1.3.1 Globcover 2009

```r
path <- fp(p_dl, "globcover_2009")
r <- raster(fp(path, "GLOBCOVER_L4_200901_200912_V2.3.tif"))
r <- crop(r, y = sa_buf_gcs, overwrite = TRUE,
          filename = fp(path, "globSA.tif"))  # crop
gdalwarp(srcfile = filename(r), overwrite = TRUE,
         dstfile = fp(path, "globSA_alb.tif"), t_srs = prjstr,
         tr = c(300, 300))
r <- raster(fp(path, "globSA_alb.tif"))
```

3.1.3.2 MODIS landcover

```r
path <- fp(p_dl, "MCD12Q1/")

# set up tiles we need and list of corresponding downloads
```

```r
tiles <- expand.grid("h" = c(19, 20), "v" = c(11, 12))
tiles <- apply(tiles, 1, function(x) paste("h", x[1], "v", x[2], sep = ""))
url <- "https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.051/2011.01.01/"
tile.names <- getURL(url, verbose=TRUE, dirlistonly = TRUE)
tile.names1 <- strsplit(tile.names, "alt")[[1]]
tile.names2 <- unique(substr(tile.names1, 20, 64))
modnames <- tile.names2[sapply(tiles, function(x) grep(x, tile.names2))]

# Quick convert of one MODIS set to get funny resolution
# have to first change installation of gdal, because homebrew version of gdal
# 1.11.5 doesn't have hdf support
gdpath <- "/Library/Frameworks/GDAL.framework/Versions/2.1/Programs"
gdal_setInstallation(search_path = gdpath, rescan = TRUE)
gdal_translate(src_dataset = fp(path, modnames[1]),
               dst_dataset = fp(path, "modis1.tif"), sd_index= 1)
r <- raster(fp(path, "modis1.tif"))  # picks up MODIS resolution

# Translate to geotiff
batch_gdal_translate(infiles = fp(path, modnames), outdir = path,
                     outsuffix = "_LC.tif", sd_index = 1)
sdnames <- dir(path, pattern = paste0("LC.tif"), full.names = TRUE)

# set gdal back to 1.11.5, for consistency
gdpath <- "/usr/local/bin"
gdal_setInstallation(search_path = gdpath, rescan = TRUE)
getOption("gdalUtils_gdalPath")[[1]]$path

# warp to albers, mosaic, and clip to SA extent
gdalwarp(srcfile = sdnames, overwrite = TRUE,
         dstfile = fp(path, "modis_type_1_alb.tif"),
         t_srs = prjstr, tr = res(r), te = bbox(sa_buf)[1:4])
r <- raster(fp(path, "modis_type_1_alb.tif"))

# Now check against original MODIS derivative
# r2 <- raster(fp(p_orig, "upstream/modis_type_1_alb.tif"))
# rdif <- r - r2
# rdif <- Which(rdif != 0)
# rdifnna <- Which(!is.na(rdif))
# cellStats(rdif, sum)  # 0 differences
# cellStats(rdifnna, sum)  # out of 10668451 pixels
# plot(rdif)
# identical
```

3.1.3.3 South Africa's landcover dataset (30 m)

```r
path <- fp(p_dl, "sa_lc_2009/")
lcnm <- c("sa_ag.tif", "sa_af.tif")
gdal_translate(src_dataset = fp(path, "landcover"),
               dst_dataset = fp(path, "sa_lc_2009.tif"),
               of = "GTiff", ot = "BYTE")

# process masks--extract extract landcover class 2, cultivation, and class 6,
# plantations, resample, mask with gdal tools
```

```r
av <- c(2, 6)
a <- "sa_lc_2009.tif"
dang <- Sys.time()
r <- lapply(1:2, function(j) {  # j <- 1
  print(paste("extracting cover for", lcnm[j]))
  gdal_calc(cstr = paste("A==", av[j], sep = ""), x = list("A" = fp(path, a)),
            type = "Byte", filename = fp(path, lcnm[j]))
  nm <- fp(p_data, paste0(gsub("\\.tif", "", lcnm[j]), "_masked.tif"))
  ext <- sapply(c("xmin", "ymin", "xmax", "ymax"), function(x) {
    slot(extent(sa_r), x)
  })
  print(paste("warping and masking", gsub(path, "", nm)))
  gdalwarp(srcfile = fp(path, lcnm[j]), t_srs = projection(sa_r),
           dstfile = nm,
           r = "average", ot = "Float32", te = ext, srcnodata = 255,
           dstnodata = 255, tr = c(1000, 1000), of = "GTiff", verbose = TRUE,
           overwrite = TRUE)
  file.remove(fp(path, lcnm[j]))
  r <- raster(nm)
})
Sys.time() - dang
```

3.1.4 Mask out sugarcane

This draws on a proprietary landcover dataset supplied by GeoTerraImage, who should be contacted for access.

```r
path <- fp(p_dl, "kzn_landcover/")
unzip("Clp_KZN_2011_V1_grid_w31.zip", exdir = path)
gdal_translate(src_dataset = fp(path, "Clp_KZN_2011_V1_grid_w31/kznlc11v1w31"),          dst_data
system(paste("rm -r", fp(path, "Clp_KZN_2011_V1_grid_w31")))
r <- raster(fp(path, "kznlandcover.tif"))

# system call to gdal_calc.py to extract landcover class 2, cultivation
nm <- fp(path, "kzn_cane.tif")  # file.remove(kznlconm)
ss <- strsplit(filename(r), "/")
nnm <- ss[[1]][length(ss[[1]])]
pcalc <- paste0("gdal_calc.py -A ", filename(r), " --outfile=", nm,
                " --overwrite --calc='(A==9)|(A==10)'")
system.time(system(pcalc))
r <- raster(nm)

# Warp and resample to SA grid
gdalwarp(srcfile = filename(r), t_srs = projection(sa_r),
         dstfile = fp(path, "kzn_cane_1km.tif"), r = "average", ot = "Float32",
         srcnodata = 255, tr = c(1000, 1000), of = "GTiff",
         verbose = TRUE, overwrite = TRUE)
r <- raster(fp(path, "kzn_cane_1km.tif"))
r <- resample(r, sa_r, method = "bilinear")
r <- mask(r, sa_r, filename = fp(path, "kzn_cane_masked.tif"),
          overwrite = TRUE)
```

3.1.5 MODIS cropland variants

Different variants of cropland fractions from MODIS classes 12 and 14 (from the IGBP-DIS scheme). Class 14 is a cropland mosaic class, with minimum of 10% and a max of 60%, so create variants of 10%, 35% (mean) and 60%.

```r
# MODIS
path <- fp(p_dl, "MCD12Q1/")
r <- raster(fp(path, "modis_type_1_alb.tif"))
r <- stack(r == 12, r == 14)
r <- brick(r)
names(r) <- c("cropland", "mosaic")

# assign percentages
minset <- c(100, 10)  # assume 10% is min crop cover for MODIS mosaic class
meanset <- c(100, 35)  # assume 35% is mean crop cover
maxset <- c(100, 60)  # assume 60% is mean crop cover
r_min <- assignLCPct(r, minset, fname = fp(p_data, "mod_1_min.tif"))
r_mu <- assignLCPct(r, meanset, fname = fp(p_data, "mod_1_mu.tif"))
r_max <- assignLCPct(r, maxset, fname = fp(p_data, "mod_1_max.tif"))

# aggregate, resample, mask to SA grid
r <- lapply(list(r_min, r_mu, r_max), function(x) {
  nm <- gsub("//.tif", "", filename(x))
  print(paste("Processing", nm))
  print("...aggregating")
  agg <- aggregate(x, fact = 2, na.rm = TRUE)
  print("...resampling")
  crop1km <- resample(agg, sa_r, method = "bilinear")
  print("...masking")
  crop1km <- mask(crop1km, mask = sa_r, filename = paste0(nm, "_1kmrect.tif"),
                  overwrite = TRUE)
  file.remove(filename(x))
  return(crop1km)
})

# And them create single raster with summed proportions
r <- lapply(r, function(x) {
  nm <- paste0(gsub("\\.tif", "", filename(x)), "_sum.tif")
  paste(nm)
  o <- calc(x, sum, filename = nm, overwrite = TRUE)
  file.remove(filename(x))
  o
})
```

3.1.6 GlobCover cropland variants

And the same for GlobCover. We want Globcover class 11 (irrigated cropland), class 14 (rainfed), 20 (50-70% cropland), and 30 (20-50% cropland). For the latter two classes we create variants assumung the minimum, mean, and max of each class

```r
path <- fp(p_dl, "globcover_2009")
r <- raster(fp(path, "globSA_alb.tif"))
l <- lapply(c(11, 14, 20, 30), function(x) r2 <- r == x)  # pull out classes
```

```r
s <- stack(l)
b <- brick(s, filename = fp(path, "globSA_crop.tif"), overwrite = TRUE)
rm(s, l)

# assign percentages
minset <- c(100, 100, 50, 20)
maxset <- c(100, 100, 70, 50)
meanset <- c(100, 100, 60, 35)
r_min <- assignLCPct(b, minset, fname = fp(p_data, "globSA_300_min.tif"))
r_mu <- assignLCPct(b, meanset, fname = fp(p_data, "globSA_300_mu.tif"))
r_max <- assignLCPct(b, maxset, fname = fp(p_data, "globSA_300_max.tif"))

# Aggregate them and resample and mask to SA grid
r <- lapply(list(r_min, r_mu, r_max), function(x) {
  nm <- gsub("_300|.tif", "", filename(x))
  print(paste("Processing", nm))
  print("...aggregating")
  agg <- aggregate(x, fact = 3, na.rm = TRUE)  # equiv to summing, div by 900
  print("...resampling")
  crop1km <- resample(agg, sa_r, method = "bilinear")
  print("...masking")
  crop1km <- mask(crop1km, mask = sa_r, filename = paste0(nm, "_1kmrect.tif"),
                  overwrite = TRUE)
  file.remove(filename(x))
  return(crop1km)
})

# And them create single raster with summed proportions
r <- lapply(r, function(x) {  # x <- r[[1]]
  nm <- paste0(gsub("\\.tif", "", filename(x)), "_sum.tif")
  paste(nm)
  o <- calc(x, sum, filename = nm)
  file.remove(filename(x))
  return(o)
})
```

3.1.7 Further masking of reference maps

```r
# Bring in SA landcover sets and mask
cover2007 <- brick(spathfunc("cover2007.tif"))
cover2011 <- brick(spathfunc("cover2011.tif"))

# sum these also with and without horticulture
sumfun <- function(x) sum(x, na.rm = TRUE)
r <- lapply(list(cover2007, cover2011), function(y) {
  r <- calc(y[[-6]], sumfun)
  r[r > 100] <- 100
  fnm <- fp(p_data, gsub("\\.tif", "sum.tif", basename(filename(y))))
  writeRaster(r, fnm, overwrite = TRUE)
})

# calculate areas
```

```r
cellStats(cover2011[[2]] / 100, sum) /
  cellStats(calc(cover2011 / 100, sumfun), sum) * 100  # communal areas 18.7\% cropland
cellStats(cover2011[[6]] / 100, sum) /
  cellStats(calc(cover2011 / 100, sumfun), sum) * 100  # hort 3.1\%
# Remove communal farmlands from both dates
sust_mask <- !is.na(cover2007[[2]]) | !is.na(cover2011[[2]])
sust_mask[sust_mask > 0] <- NA

# Set up SA mask including sugarcane and forestry -- add in Mpumalanga sugar
# cane when it becomes available
m1 <- raster(fp(p_dl, "kzn_landcover/kzn_cane_masked.tif"))
m1[is.na(m1)] <- 0
m1[m1 > 0] <- NA
m2 <- raster(spathfunc("sa_af_masked.tif"))
m2[is.na(m2)] <- 0
m2[m2 > 0] <- NA
sa_masks <- sa_r > 0
sa_masks <- m1 + m2 + sa_masks
rm(m1, m2)
full_mask <- sa_masks + sust_mask

# Apply masks
r <- lapply(r, function(x) {
  out_name <- paste0(gsub("\\.tif", "", filename(x)), "_mask.tif")
  o <- mask(x, full_mask, file = out_name, overwrite = TRUE)
  file.remove(filename(x))
  return(o)
})
```

4 Calculatiing cropland bias and accuracy

This section of the analysis calculates the differences between the reference cropland percentages and those from the various landcover products. It draws from results generated by the landcover pre-processing. Bias statistics calculated here are primarily for the supplementals.

4.1 Analyses

```r
library(lmisc)
library(raster)
library(croplandbias)
library(xtable)
p_root <- proj_root("croplandbias")
p_data <- fp(p_root, "croplandbias/inst/extdata/landcover")
```

4.1.1 Load cropland test grids

Derived from global landcover products

```r
salc <- raster(spathfunc("sa_ag_masked.tif")) * 100
nms <- paste0("globSA_", c("min", "mu", "max"), "_1kmrect_sum.tif")
```

```r
globsa_list_sum <- lapply(nms, function(x) raster(spathfunc(x)))
nms <- paste0("mod_1_", c("min", "mu", "max"), "_1kmrect_sum.tif")
modsa_list_sum <- lapply(nms, function(x) raster(spathfunc(x)))
glc <- raster(spathfunc("glcsa_masked.tif"))
lclist <- c(salc, globsa_list_sum, modsa_list_sum, glc)  # into list
names(lclist) <- c("sa30", "globmin", "globmu", "globmax", "modmin", "modmu",
                   "modmax", "glc")
```

4.1.1.1 Reference data

```r
data(sashp)
nms <- paste0("cover", c("2007", "2011"), "sum_mask.tif")
gti <- lapply(nms, function(x) raster(spathfunc(x)))
names(gti) <- c("g2007", "g2011")
```

4.1.1.2 Apply NA masks

```r
sumna <- function(x) sum(x, na.rm = FALSE)
namask <- calc(stack(stack(gti), stack(lclist)), sumna)
namask[namask >= 0] <- 1
namask2 <- !is.na(namask)  # set NAs to zero
# cellStats(namask2, sum)
namask[namask == 0] <- NA
# cellStats(namask, sum)
writeRaster(namask, filename = fp(p_data, "namask.tif"),
            overwrite = TRUE)

# mask out all joint NAs
gti <- lapply(gti, function(x) mask(x, namask))
lclist <- lapply(lclist, function(x) mask(x, namask))
cropl <- c(gti$g2011, lclist[c("sa30", "globmu", "modmu", "glc")])
names(cropl)[1] <- "gti"
save(cropl, file = fp(p_data, "cropland-1km.rda"))
```

4.1.2 Calculate differences at different levels of aggregation

This step draws on functions I wrote to calculate the actual and absolute differences between a given "master" grid and lists of other grids. This comparison is done at multiple aggregations, ranging from 1-600 km resolution cropland averages. The lists have two levels of nesting, here the upper provides the resolution, the inner provides the different landcover datasets.

```r
# Aggregate rasters
fact <- c(5, 10, 25, 50, 100)
#fact <- 25

area_wgts <- aggregate_rast_list(fact, list(namask2), fun = sum)  # cells/pixel
lc_agg <- aggregate_rast_list(fact, lclist)   # landcover rasters
gti_agg <- aggregate_rast_list(fact, gti)  # GTI rasters

# Calculate differences
# actual
dlist_act <- rast_list_math(list(names(lclist), names(lc_agg), names(gti)),
                             gti_agg, lc_agg, "a - b", silent = FALSE)
# absolute
dlist_abs <- rast_list_math(list(names(lclist), names(lc_agg), names(gti)),
                             gti_agg, lc_agg, "abs(a - b)", silent = FALSE)
dlist_gti <- rast_list_math(list("g2007", names(gti_agg), "g2011"),
                             gti_agg, gti_agg, "a - b", silent = FALSE)


save(lc_agg, gti_agg, area_wgts, file = fp(p_data, "agg_cropland.rda"))
save(dlist_act, file = fp(p_data, "d_grid_act.rda"))
save(dlist_abs, file = fp(p_data, "d_grid_abs.rda"))

# Write out the gti cover raster and the actual bias rasters for running impact
# of bias examples in land surface model (25 km resolution)
r <- gti_agg$f25$g2011
projection(r) <- projection(salc)
writeRaster(r, filename = fp(p_data, "gti_2011_25km.tif"), overwrite = TRUE)
b <- brick(stack(lapply(c("sa30", "globmu", "modmu", "glc"), function(x) {
  dlist_act[[x]]$f25$g2011
})))
projection(b) <- projection(salc)
writeRaster(b, filename = fp(p_data, "lc_bias_25km.tif"), overwrite = TRUE)

# r <- raster(fp(p_data, "gti_2011_25km.tif"))
# lcb <- brick(fp(p_data, "lc_bias_25km.tif"))
# plot(r - lcb, axes = FALSE, box = FALSE)
# plot(lcb)
# plot(round(lc_agg$f25$glc - (r - lcb$lc_bias_25km.4), 7))
```

4.1.3 Error versus cropland density

4.1.3.1 Weighting rasters

Based on number of non-NA cells in each aggregated pixel, and then "bin" rasters, where each category represents a different cropland fractional cover range (0-5%, 5-10%, etc)

```r
# cropland cover bins
binv <- seq(0, 100, 5)
gti_bins <- lapply(gti_agg, function(x) {
  lapply(x, function(y) cut(y, breaks = binv, include.lowest = TRUE))
})
```

4.1.3.2 Calculate bias/MAE, pooling MODIS and GlobCover across all versions

Step 1. Reshape lists, so that all MODIS and GlobCover variants are pooled by scale

```r
# Set up indexing parameters
fact_all <- c(1, fact)# full aggregation levels
lev_vec <- paste("f", fact_all, sep = "")
nm_rt <- c("sa30", "glob", "mod", "glc")

# Reshape the list of actual landcover differences
dlists <- lapply(list(dlist_act, dlist_abs), function(i) {
  resh <- lapply(nm_rt, function(x) {  # select sensor
    l2 <- lapply(lev_vec, function(y) {  # isolate by scale
      l3 <- lapply(i[grep(x, names(i))], function(z) z[[y]])  # sensor by scale
        named_out(l3, names(i)[grep(x, names(i))])
      })
    named_out(l2, lev_vec)
  })
  named_out(resh, nm_rt)
})
names(dlists) <- c("act", "abs")
rm(dlist_act, dlist_abs)  # remove original to make space
```

4.1.3.3 Then calculate bias/MAE across datasets

Note: this uses an older and slower version of the code. Subsequent analyses use the faster version based on data.tables.

```r
dang <- Sys.time()
# i <- dlists[[1]]; x <- names(i)[1]; y <- lev_vec[5]; j <- 14; z <- 1; k <- 1
dstats_all <- lapply(dlists, function(i) {
  print("processing 1 of 2 main bias lists")
  dstats <- lapply(names(i), function(x) {
    print(paste("..", x))
    l1 <- lapply(lev_vec, function(y) {
      print(paste("....", y))
      d1 <- do.call(rbind, lapply(1:(length(binv) - 1), function(j) {
        print(paste("......bin", j))
        d2 <- do.call(rbind, lapply(1:length(i[[x]][[y]]), function(z) {
          print(paste("........variant", z))
          l3 <- i[[x]][[y]][[z]]
          d3 <- do.call(rbind, lapply(1:length(l3), function(k) {
            rs <- gti_bins[[y]][[k]] == j
            rs[rs == 0] <- NA
            o <- rs * l3[[k]]
            w <- area_wgts[[y]][[1]] * rs
            oo <- cbind.data.frame("lc" = rep(names(i)[z], length(w)),
                                   getValues(o), getValues(w))
          }))
          d3f <- d3[!is.na(d3[, 2]), ]
        }))
        if(nrow(d2) > (length(i[[x]][[y]]) * 10)) {
          qs <- c(nrow(d2), box_stats(d2[, 2], weighted = TRUE,
                                      weight.opts = list("weights" = d2[, 3])))
        } else if((nrow(d2) > 0) & (nrow(d2) < length(i[[x]][[y]]) * 20)) {
          qs <- c(nrow(d2), rep(NA, 5),
                  Hmisc::wtd.mean(d2[, 2], weights = d2[, 3], na.rm = FALSE))
        } else {
          qs <- rep(NA, 7)
        }
      }))
      cbind(binv[-1], d1)
    })
    named_out(l1, lev_vec)
  })
  named_out(dstats, names(i))
})
dut <- Sys.time() - dang
save(dstats_all, file = fp(p_data, "bias_stats.rda"))
```

4.1.4 Differences between 2007 and 2011 reference

```r
dstats_gti <- lapply(lev_vec, function(x) {
    #x <- lev_vec[[2]]
    i <-  dlist_gti$g2007[[x]][[1]]
    print(paste("..", x))
    d1 <- do.call(rbind, lapply(1:(length(binv) - 1), function(y) {
      #y <- 1
      print(paste("....", y))
      rs <- gti_bins[[x]][[2]] == y
      rs[rs == 0] <- NA
      o <- rs * i
      w <- area_wgts[[x]][[1]] * rs
      oo <- cbind.data.frame(rep(y, length(w)), getValues(o), getValues(w))
      oof <- oo[!is.na(oo[, 2]), ]
      if(nrow(oof) > 10) {
        qs <- c(nrow(oof), box_stats(oof[, 2], weighted = TRUE,
                                     weight.opts = list("weights" = oof[, 3])))
      } else if((nrow(oof) > 0) & (nrow(oof) < 10)) {
        qs <- c(nrow(oof), rep(NA, 5),
                Hmisc::wtd.mean(oof[, 2], weights = oof[, 3], na.rm = FALSE))
      } else {
        qs <- rep(NA, 7)
      }
    }))
    cbind(binv[-1], d1)
})
save(dstats_gti, file = fp(p_data, "bias_stats_gti.rda"))
```

Checks: compared number of non-NA cells coming out of each GTI version at each level of aggregation (fine); compared cell counts between previous committed version and this one (difference exists, but due to fact that gtimu is no longer considered; checked that previous versions of difference rasters and reshapes of them were referencing the correct GTI-landcover bias rasters (i.e. to make sure rasters weren't incorrectly indexed: all fine)); box_stats with weighted quantile option compared to previous commit where wtd.quantile was used separately (fine). Compared results from original roughed out version of approach for calculating bias statistics with results from new outputs list (using GLC-Share at 20 km aggregation as the comparison)

4.1.5 2011 reference-test bias/accuracy

Whole country and agricultural areas only

```r
load(fp(p_data, "d_grid_act.rda"))  # actual diffence grids

# Subset difference grids, for 2011
snms <- c("sa30", "globmu", "modmu", "glc")
lc_pct_diff <- lapply(dlist_act[snms], function(x) {
  sapply(x, function(y) list(y$g2011))
})

# construct agricultural area union mask
lev <- names(lc_pct_diff[[1]])
lcu <- lapply(lev, function(x) {
  print(paste(".", x))
  lcb <- lapply(snms, function(y) {
    print(paste("...", y))
    gti_gt0 <- Which(gti_agg[[x]][[1]] > 0)
    lc_gt0 <- Which(lc_agg[[x]][[y]] > 0)
    all_gt0 <- gti_gt0 + lc_gt0
    all_gt0[all_gt0 > 0] <- 1
    all_gt0
  })
  named_out(lcb, snms)
})
names(lcu) <- lev

bins <- lapply(gti_bins, function(x) x$g2011)
awgts <- lapply(area_wgts, function(x) x[[1]])

# calculate means for 2011
wm <- function(x, w) stats::weighted.mean(x, w)
wma <- function(x, w) stats::weighted.mean(abs(x), w)
a <- bias_stats_list(bins, awgts, lcu, lc_pct_diff, wm, "mu", "bias", TRUE)
b <- bias_stats_list(bins, awgts, lcu, lc_pct_diff, wma, "mua", "bias", TRUE)
mu_dt <- rbind(a, b)
mu <- extract_stat(lev, snms, "all", "mu", "bias", mu_dt)  # ag area means
mua <- extract_stat(lev, snms, "all", "mua", "bias", mu_dt)  # ag area abs mean
mu_nm <- extract_stat(lev, snms, "all", "mu.nm", "bias", mu_dt)  # country mean
mua_nm <- extract_stat(lev, snms, "all", "mua.nm", "bias", mu_dt)  # country abs
# across bin mean and abs mean
mu_bin <- mu_dt[!like(bin, "all") & bvals == "mu", mean(bias), by = .(ol, il)]
setnames(mu_bin, "V1", "bias")
mua_bin <- mu_dt[!like(bin, "all") & bvals == "mua", mean(bias),
                 by = .(ol, il)]
setnames(mua_bin, "V1", "bias")
# mu_dt[il == "sa30" & bin != "all" & ol == "f1" & bvals == "mua", mean(bias)]

# check
# Older variant of code from compare-landcover.Rmd to evaluate whether newer DT
# version is finding correct results
bv <- "mua"
for(chk in list(c("modmu", "f25"), c("glc", "f10"), c("sa30", "f50"),
                c("globmu", "f1"))) {
  x <- chk[1]
  y <- chk[2]
  print(paste(".", x, "..", y))
  rs <- lcu[[y]][[x]]
  rs[rs == 0] <- NA
  l3 <- abs(lc_pct_diff[[x]][[y]])
  o <- rs * l3
  w <- awgts[[y]][[1]] * rs
  wmu <- weighted.mean(getValues(o), getValues(w), na.rm = TRUE)
  print(b[ol == y & il == x & bvals == bv & bin == "all", bias] ==
          round(wmu, 2))
}  # check

svec <- c("mu_nm", "mua_nm", "mu", "mua", "mu_bin", "mua_bin")
aa <- c(rep("Country", 2), rep("Agricultural", 2), rep("Density independent", 2))
bb <- c("Bias", "MAE", "Bias", "MAE", "Bias", "MAE")
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")

lcf_out11 <- do.call(rbind, lapply(1:length(svec), function(i) {
  ao <- do.call(rbind, lapply(snms, function(x) {
    ai <- sapply(lev, function(y) {
      get(svec[i])[ol == y & il == x, bias]
    })
    named_out(ai, paste(c(1, fact), "km"))
  }))
  #cbind.data.frame("Region" = aa[i], "Map" = mcap, a)
  cbind.data.frame(Region = aa[i], Metric = bb[i], "Map" = mcap, ao)
}))

caption <- paste("Biases and mean absolute errors (MAE) in cropland maps",
                 "relative to the 2011 reference map for each aggregation",
                 "scale calculated over the entire country, for the union of",
                 "agricultural regions (cropland $>$ 0), and as density",
                 "independent means, wherein the mean bias/MAE values",
                 "for each of 20 cropland cover classes (representing",
                 "5\\% increments of cover 0\\% to 100\\% defined",
                 "by the reference map) were calculated and then averaged.")
lcfout_xtab <- xtable(lcf_out11, digits = 1, caption = caption)
print(lcfout_xtab, type = "latex",
      file = fp(p_root,
               "croplandbias/inst/paper/figures/cropland2011-bias.tex"),
      tabular.environment = "longtable", floating = FALSE,
      caption.placement = "top", include.rownames = FALSE)

# do.call(rbind, lapply(snms, function(i) {
#   mu_dt[!like(bin, "all") & ol == "f1" & il == i & bvals == "mu.nm", N]
# }))
mubias_2011 <- mu_dt
save(lcf_out11, mubias_2011, file = fp(p_data, "lcf_out2011.rda"))
```

4.1.6 2007 reference-test bias/accuracy

Whole country and agricultural areas only

```
# Subset difference grids, for 2011
snms <- c("sa30", "globmu", "modmu", "glc")
lc_pct_diff <- lapply(dlist_act[snms], function(x) {
  sapply(x, function(y) list(y$g2007))
})

# construct agricultural area union mask
```

```r
lev <- names(lc_pct_diff[[1]])
lcu <- lapply(lev, function(x) {
  print(paste(".", x))
  lcb <- lapply(snms, function(y) {
    print(paste("...", y))
    gti_gt0 <- Which(gti_agg[[x]][[1]] > 0)
    lc_gt0 <- Which(lc_agg[[x]][[y]] > 0)
    all_gt0 <- gti_gt0 + lc_gt0
    all_gt0[all_gt0 > 0] <- 1
    all_gt0
  })
  named_out(lcb, snms)
})
names(lcu) <- lev

bins <- lapply(gti_bins, function(x) x$g2007)
awgts <- lapply(area_wgts, function(x) x[[1]])

# calculate means for 2007
wm <- function(x, w) stats::weighted.mean(x, w)
wma <- function(x, w) stats::weighted.mean(abs(x), w)
a <- bias_stats_list(bins, awgts, lcu, lc_pct_diff, wm, "mu", "bias", TRUE)
b <- bias_stats_list(bins, awgts, lcu, lc_pct_diff, wma, "mua", "bias", TRUE)
mu_dt <- rbind(a, b)
mu <- extract_stat(lev, snms, "all", "mu", "bias", mu_dt)
mua <- extract_stat(lev, snms, "all", "mua", "bias", mu_dt)
mu_nm <- extract_stat(lev, snms, "all", "mu.nm", "bias", mu_dt)
mua_nm <- extract_stat(lev, snms, "all", "mua.nm", "bias", mu_dt)
# across bin mean and abs mean
mu_bin <- mu_dt[!like(bin, "all") & bvals == "mu", mean(bias), by = .(ol, il)]
setnames(mu_bin, "V1", "bias")
mua_bin <- mu_dt[!like(bin, "all") & bvals == "mua", mean(bias), by = .(ol, il)]
setnames(mua_bin, "V1", "bias")

# check
# Older variant of code from compare-landcover.Rmd to evaluate whether newer DT
# version is finding correct results
bv <- "mua"
for(chk in list(c("modmu", "f25"), c("glc", "f10"), c("sa30", "f50"),
                c("globmu", "f1"))) {
  x <- chk[1]
  y <- chk[2]
  print(paste(".", x, "..", y))
  rs <- lcu[[y]][[x]]
  rs[rs == 0] <- NA
  l3 <- abs(lc_pct_diff[[x]][[y]])
  o <- rs * l3
  w <- awgts[[y]][[1]] * rs
  wmu <- weighted.mean(getValues(o), getValues(w), na.rm = TRUE)
  print(b[ol == y & il == x & bvals == bv & bin == "all", bias] ==
          round(wmu, 2))
}  # check
```

```r
svec <- c("mu_nm", "mua_nm", "mu", "mua", "mu_bin", "mua_bin")
aa <- c(rep("Country", 2), rep("Agricultural", 2),
        rep("Density independent", 2))
bb <- c("Bias", "MAE", "Bias", "MAE", "Bias", "MAE")
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")

lcf_out07 <- do.call(rbind, lapply(1:length(svec), function(i) {
  ao <- do.call(rbind, lapply(snms, function(x) {
    ai <- sapply(lev, function(y) {
      get(svec[i])[ol == y & il == x, bias]
    })
    named_out(ai, paste(c(1, fact), "km"))
  }))
  #cbind.data.frame("Region" = aa[i], "Map" = mcap, a)
  cbind.data.frame(Region = aa[i], Metric = bb[i], "Map" = mcap, ao)
}))

caption <- paste("Biases and mean absolute errors (MAE) in cropland maps",
                 "relative to the 2007 reference map for each aggregation",
                 "scale",
                 "calculated over the entire country, for the union of",
                 "agricultural regions, and as density",
                 "independent means, wherein the mean bias/MAE values",
                 "for each of 20 cropland cover classes",
                 "(representing 5\\% increments of cover 0\\% to 100\\%",
                 "defined",
                 "by the reference map) were calculated and then averaged.")
lcfout_xtab <- xtable(lcf_out07, digits = 1, caption = caption)
print(lcfout_xtab, type = "latex",
      file = fp(p_root,
                paste0("croplandbias/inst/paper/figures/",
                       "cropland2007-bias.tex")),
      tabular.environment = "longtable", floating = FALSE,
      caption.placement = "top", include.rownames = FALSE)
```

5 Cropland bias/accuracy, part 2

The following provide further analyses and plots of bias and accuracy metrics calculated from spatial error maps created in cropland-1. ## Spatial biases ### Datasets

```r
library(croplandbias)
library(RColorBrewer)

# Paths
p_root <- proj_root("croplandbias")
p_data <- fp(p_root, "croplandbias/inst/extdata/landcover")
p_fig <- fp(p_root, "croplandbias/inst/paper/figures/")

# Load in datasets
data(sashp)  # SA shape
load(spathfunc("d_grid_act.rda"))  # actual diffence grids
load(spathfunc("d_grid_abs.rda"))  # actual diffence grids
load(spathfunc("bias_stats.rda"))  # bias statistics
load(spathfunc("bias_stats_gti.rda"))  # bias statistics
load(spathfunc("agg_cropland.rda"))  # aggregated cropland
# cellStats(!is.na(dlist_act$modmu$f1$g2011), sum)
```

5.1 Error maps

Plot maps of biases at different resolutions. To avoid the visual distortion of NA areas filled in by larger levels of aggregation, each aggregated difference must be disaggregated first and masked before plotting.

```
namask <- raster(spathfunc("namask.tif"))
# cellStats(namask, sum)
snms <- c("sa30", "globmu", "modmu", "glc")  # all lc data (no MOD/GC extremes)
lev <- c("f1", "f25", "f50", "f100")  # just show four levels

# disggregate selected rasters at selected levels
# x <- snms[3]; y <- lev[3]
disagg <- lapply(snms, function(x) {
  l1 <- lapply(lev, function(y) {
    if(y == "f1") {
      r <- dlist_act[[x]][[y]]$g2011
    } else {
      r <- disaggregate(dlist_act[[x]][[y]]$g2011,
                        fact = as.numeric(gsub("f", "", y)))
      r <- raster::mask(crop(r, namask), namask)
    }
  })
  named_out(l1, lev)
})
names(disagg) <- snms

# writeRaster(disagg$globmu$f100,
#             filename = "external/ext_data/test/disagg100km.tif")
```

5.1.1 Plot of selected rasters and levels

```
# Plotting colors
# cols <- colorRampPalette(c("red", "grey80", "blue4"))
brks <- c(-100.1, -75, -50, -25, -10, -5, -1, 1, 5, 10, 25, 50, 75, 100.1)
cols <- c(rev(brewer.pal(length(brks) / 2, "Reds")[-1]), "grey80",
          brewer.pal(length(brks) / 2, "Blues")[-1])
brklen <- length(brks) - 1
# cols <- cols(brklen)
# cols[c(1, length(cols))] <- c("darkred", "purple3")
legtext <- "% Difference"
cx <- 1.4
lcol <- "black"
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
lev2 <- c("1 km", "25 km", "50 km", "100 km")
pdf(fp(p_fig, "fig1a.pdf"), height = 7, width = 7)
par(mfrow = c(4, 4), mar = c(0, 0, 0, 0), oma = c(3.7, 1.5, 1.5, 0))
```

```r
for(i in 1:length(snms)) {
  print(snms[i])
  for(j in 1:length(lev)) {
    print(lev[j])
    plot(sashp, lty = 0)
    plot(disagg[[snms[i]]][[lev[j]]], add = TRUE, col = cols, breaks = brks,
         legend = FALSE)
    if(j == 1) mtext(mcap[i], side = 2, line = -0.25, cex = cx)
    if(i == 1) mtext(lev2[j], side = 3, line = -0.5, cex = cx)
  }
}
flex_legend(ncuts = brklen, legend.text = legtext,
            legend.vals = round(brks), longdims = c(0.2, 0.8),
            shortdims = c(0.05, 0.01), colvec = cols, srt = c(270, 0),
            horiz = TRUE, textside = "bottom", legend.pos = c(4, 7),
            leg.adj = list(c(0.2, 0.35), c(0.5, -0.5)), cex.val = cx,
            textcol = lcol, bordercol = lcol)
dev.off()

lev <- names(dlist_abs$sa30)
```

The map below was created by the code above, showing biases at 1 km, 25 km, 50 km, and 100 km from South Africa's landcover product, the mean estimates from GlobCover and MODIS, and GLC-SHARE.

5.2 Bias/MAE in relation to field density

5.2.1 1 km Bias/MAE with boxplots

For supplemental.

```r
#btype <- c("bias_act.pdf", "bias_abs.pdf")
# plot matrix
nc <- 4
m <- matrix(rep(c(rep(1, 8), rep(2, 8), rep(3, 8), rep(4, 8)), 2), nrow = 6,
            ncol = 32, byrow = TRUE)
cols <- c("red", "orange3", "green4", "blue", "yellow")
bcol <- c("grey50", "grey50", "grey50", "grey")
lcnms <- c("SA LC", "GlobCover", "MODIS", "GLC-Share")
fcol <- "grey"
lev <- names(dstats_all$act$sa30)
lvec <- seq(5, 100, 5)
pp <- list("x" = c(-2, 21), "y" = c(0, 0),
           "x21" = list(c(0, 9), c(9, 21)),
           "y2" = list(c(seq(-100, -10, 10), seq(10, 100, 10)),
                       seq(10, 100, 10)),
           "y1lwd" = list(2, 1),
           "y2lwd" = 0.5, #c(1, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 1, 0.5, 1),
           "y2lc" = list(c("black", rep("grey80", 4), "black",
                           rep("grey80", 4), rep("grey80", 4),
                           "black", rep("grey80", 4),"black"),
                         rep("grey80", 10)),
           "yl" = list(c(-100, 100), c(0, 100)),
           "yl2" = c(-20, 60), "xl" = c(1, 20),
           "xaxl" = unlist(lapply(seq(5, 100, 10), function(x) c(x, " "))),
           "xaxl2" = ifelse(seq(5, 100, 5) %in% c(25, 50, 75), seq(5, 100, 5),
                            " "),
           "yax1" = list(seq(-100, 100, 50), seq(0, 100, 20)),
           "ycol" = c("grey70"))  # plot parameters

pdf(fp(p_fig, "biases_1km.pdf"), width = 7, height = 5)
lmat <- rbind(m, m + max(m))
l <- layout(lmat)
par(mar = c(3, 0, 0, 0), oma = c(1, 4, 2, 2))
for(bt in 1:2) {
  snms <- names(dstats_all[[bt]])
  dset <- dstats_all[[bt]][snms]
  for(i in 1:length(snms)) { i
    d <- dset[[snms[i]]]$f1
    n <- nrow(d)
    plot(pp$x, pp$y, ylim = pp$yl[[bt]], xlim = pp$xl, type = "l",
         lwd = pp$y1lwd[[bt]], axes = FALSE,
         xlab = "", ylab = "")
    if(i == 1) axis(2, at = pp$yax1[[bt]], labels = pp$yax1[[bt]], las = 2,
                    mgp = c(1, 0.6, 0))
    axis(1, at = 1:20, labels = pp$xaxl, las = 2, mgp = c(1, 0.4, 0),
         tcl = -0.3, pos = pp$yaxp[[bt]], cex = 0.8)
    for(z in 1:length(pp$y2[[bt]])) {
      lines(pp$x, rep(pp$y2[[bt]][z], 2), lwd =pp$y2lwd,
            col = pp$y2lc[[bt]][z])
    }
    for(k in 1:n) {
      a <- seq(grep("2.5", colnames(d)), by = 1, length.out = 6)
      if(!is.na(d[k, 2])) {
        boxplot_v(k, d[k, a], n, 150, 10, bcol, fcol, pcex = 0.5,
                  lwd = c(1, 0.5, 2))
      }
    }
    lines(1:n, d[, "mu"], pch = 20, col = cols[i], lwd = 3)
    if(i == 1) mtext(ifelse(bt == 1, "Bias (%)", "Mean absolute error (%)"),
                     side = 2, line = 2.2, cex = 0.8)
    if(bt == 1) mtext(lcnms[i], side = 3, line = 0, col = cols[i])
  }
}
mtext("% Cropland", side = 1, outer = TRUE, line = -1, cex = 0.8)
dev.off()
```

5.2.2 Bias/MAE with scale

Note: the mean values presented in the figure here is a straight mean across 5 percentile bins, which gives bias independent of cropland density. Supplementary tables report the bin-weighted biases.

```r
levnms <- paste(gsub("f", "", lev), "km")
m2 <- unlist(lapply(seq(1, by = 1, length = length(lev) * 4), function(x) {
  rep(x, nc)
}))
m3 <- do.call(rbind, lapply(seq(1, length(m2), length(lev) * nc), function(x) {
  do.call(rbind, rep(list(m2[x:(x + length(lev) * nc - 1)]), nc))
}))

yl <- seq(pp$yl2[1], pp$yl2[2], 10)
ylcol <- c("black", "grey", "black", rep("grey", 5), "black")
lt <- c(5, 1)
ylwd <- c(1.5, 2)

wm <- function(x, w) stats::weighted.mean(x, w, na.rm = TRUE)
wma <- function(x, w) stats::weighted.mean(abs(x), w)

pdf(fp(p_fig, "biases_1-100km.pdf"), width = 7.5, height = 7)
l <- layout(m3)
# layout.show(l)
# i <- 1; j <- 1; bt <- 1
par(mar = c(0.5, 0, 1, 0.3), oma = c(5, 5, 2, 2))
for(i in 1:length(snms)) {
  highcounter <- rep(0, length(lev))
  for(j in 1:length(lev)) {
    plot(pp$x, pp$y, ylim = pp$yl2, xlim = pp$xl, type = "l", lwd = 1,
         axes = FALSE, xlab = "", ylab = "", xaxs = "i")
    for(z in 1:length(yl)) {
      lines(pp$x, rep(yl[z], 2), lwd = pp$y2lwd, col = ylcol[z])
    }
    muv <- rep(NA, 2)
    muv2 <- rep(NA, 2)
    for(bt in 1:2) {
      d <- dstats_all[[bt]][[i]][[j]]
      n <- nrow(d)
      lines(1:n, d[, "mu"], pch = 20, col = cols[i], lwd = ylwd[bt],
            lty = lt[bt])
      muv[bt] <- round(mean(d[, 8], na.rm = TRUE)) # unweighted means
      muv2[bt] <- round(wm(d[, 8], d[, 2]))  # weighted means
      toohigh <- which(d[, "mu"] > pp$yl2[2])
      if(length(toohigh) > 0) {
        highcounter[j] <- 1
        xi <- toohigh[which.max(d[toohigh, "mu"])]
        text(xi, y = pp$yl2[2] + 4, labels = round(d[xi, "mu"]), cex = 1.1,
             col = "grey50", xpd = NA)
      }
    }
    text(5, pp$yl2[2] - 5, labels = paste(muv, collapse = " / "), cex = 1.1,
         col = "grey10", font = 4)
    text(5, pp$yl2[2] - 16, labels = paste(muv2, collapse = " / "), cex = 1.1,
         col = "grey10")
    if(j == 1) axis(2, at = yl, labels = yl, las = 2, mgp = c(1, 0.6, 0))
    if(j == 1) mtext(lcnms[i], side = 2, line = 3, cex = 1, col = cols[i])
    if(i == 1) mtext(levnms[j], side = 3, line = -0.4, cex = 1.2)
    axis(1, at = c(1, 5, 10, 15, 20), labels = rep("", 5), mgp = c(1, 1, 0),
         tcl = -0.7, lwd.ticks = 1.2, pos = pp$yl2[1])
    axis(1, at = 1:20, labels = rep("", 20), las = 2, mgp = c(1, 0.4, 0),
         tcl = -0.3, pos = pp$yl2[1])
    if(i == length(snms)) {
      axis(1, at = 1:20, labels = pp$xaxl2, las = 2, mgp = c(1, 0.8, 0),
           tcl = -0.3, pos = pp$yl2[1])
    }
  }
}
mtext("% Bias/MAE", side = 2, line = 2, cex = 0.8, outer = TRUE, adj = 0.5)
mtext("% Cropland", side = 1, outer = TRUE, line = 1.5, cex = 0.8)
par(xpd = NA)
x <- grconvertX(0.75, from = "ndc", to = "user")
y <- grconvertY(0.07, from = "ndc", to = "user")
legend(x = x, y = y, legend = c("Bias", "Mean absolute error"), lty = lt,
       lwd = ylwd, bty = "n", cex = 1.2)
dev.off()
```

5.2.3 Total error

Calculate how much cropland estimates differ between datasets

```
fact <- c(5, 10, 25, 50, 100)
nms <- paste0("cover", c("2007", "2011"), "sum_mask.tif")
gti <- lapply(nms, function(x) raster(spathfunc(x)))
names(gti) <- c("g2007", "g2011")
gti <- lapply(gti, function(x) raster::mask(x, namask))  # apply namask
snms <- c("sa30", "globmu", "modmu", "glc")  # all lc data (no MOD/GC extremes)

gti_agg <- aggregate_rast_list(fact, gti)  # GTI rasters
gti_agg11 <- lapply(gti_agg, function(x) x$g2011)

# recreate cropland percentages
# lc2007 <- lapply(snms, function(x) gti$g2007 - dlist_act[[x]]$f1$g2007)
# lc2011 <- lapply(snms, function(x) gti$g2011 - dlist_act[[x]]$f1$g2011)
# plot(lc2007[[2]] - lc2011[[2]])  # same thing, as it should be

# calculate km2 cropland in LC datasets
awgts <- lapply(area_wgts, function(x) x[[1]])
lckm2 <- do.call(rbind, lapply(1:length(lev), function(x) {  # x <- 1
  gtia <- cellStats(gti_agg11[[x]] / 100 * awgts[[x]], sum)
  lca <- sapply(snms, function(y) {
    cellStats(lc_agg[[x]][[y]] / 100 * awgts[[x]], sum)
  })
  o <- c(gtia, lca)
}))
colnames(lckm2)[1] <- "gti"
rownames(lckm2) <- lev

# mean (stays about the same)
lckmmu <- do.call(rbind, lapply(1:length(lev), function(x) {  # x <- 1
  gtia <- cellStats(gti_agg11[[x]] / 100, mean)
  lca <- sapply(snms, function(y) {
    cellStats(lc_agg[[x]][[y]] / 100, mean)
  })
  o <- c(gtia, lca)
}))
```

```r
colnames(lckmmu)[1] <- "gti"
rownames(lckmmu) <- lev

# variance declines with scale, as expected
lckmvar <- do.call(rbind, lapply(1:length(lev), function(x) {  # x <- 1
  gtia <- cellStats(gti_agg11[[x]] / 100, var)
  lca <- sapply(snms, function(y) {
    cellStats(lc_agg[[x]][[y]] / 100, var)
  })
  o <- c(gtia, lca)
}))
colnames(lckmvar)[1] <- "gti"
rownames(lckmvar) <- lev

# lckm2 <- sapply(lc_agg$f1[snms], function(x) cellStats(x / 100, sum))
# lckm2 <- sapply(lc2011, function(x) cellStats(x / 100, sum))
# refkm2 <- sapply(gti, function(x) cellStats(x / 100, sum))

# area of each GTI dataset
refkm2 <- sapply(gti_agg$f1, function(x) cellStats(x / 100, sum))
refkm2[2] / refkm2[1]  # 3.7 percent more in 2011 than 2007

# bias in total cropland relative to GTI, at 1 km scale
stats2007 <- round((refkm2[2] - lckm2[1, ]) / refkm2[2] * 100, 1)
stats2011 <- round((refkm2[1] - lckm2[2, ]) / refkm2[1] * 100, 1)

# N km^2 in country
#sapply(lc2007, function(x) cellStats(!is.na(x), sum))
# sapply(lc2011, function(x) cellStats(!is.na(x) , sum))
# sapply(gti, function(x) cellStats(!is.na(x) , sum))

totkm2 <- cellStats(!is.na(gti_agg$f1$g2011), sum)
save(stats2007, stats2011, totkm2, refkm2, lckm2,
     file = fp(p_data, "total-cropland.rda"))
```

Total square kilometers of South Africa being evaluated

```
## [1] 1079793
```

Total square kilometers of cropland according to 2007 and 2011 GTI data

```
##    g2007    g2011
## 100624.4 104303.7
```

Percent bias in total cropland area estimates relative to 2007 GTI data

```
##    gti   sa30 globmu  modmu    glc
##    0.0  -26.0   21.0   26.1   -5.7
```

And relative to 2011 GTI data

```
##    gti   sa30 globmu  modmu    glc
##   -3.7  -30.6   18.2   23.4   -9.6
```

Cropland area by dataset and scale

```
##         gti   sa30  globmu modmu    glc
## f1   104304 131390   82358 77090 110272
## f5   104304 131390  82358 77090 110272
## f10  104304 131390  82358 77090 110272
## f25  104304 131390  82358 77090 110272
## f50  104304 131390  82358 77090 110272
## f100 104304 131390  82358 77090 110272
```

5.3 Cropland bias/MAE plots

5.3.1 Primary method

Weighted by **actual** cropland percentage, and then, with aggregation, by number of pixels being aggregated.

```
# Create weights mask
fact <- c(5, 10, 25, 50, 100)
namask <- raster(spathfunc("namask.tif"))  # NA mask
# cellStats(!is.na(gti$g2011), sum)
# cellStats(namask, sum)

namask2 <- !is.na(namask)
awgts <- aggregate_rast_list(fact, list(namask2), fun = sum)
awgts <- lapply(awgts, function(x) x[[1]])

# Subset difference grids, for 2011
snms <- c("sa30", "globmu", "modmu", "glc")
err2011 <- lapply(lev, function(x) {
  sapply(dlist_act[snms], function(y) list(y[[x]]$g2011))
})
names(err2011) <- lev

# gti_agg <- aggregate_rast_list(fact, gti)  # GTI rasters
# gti_agg11 <- lapply(gti_agg, function(x) x$g2011)

# calculate bias/MAE for data
a <- bias_statsw_list(gti_agg11, awgts, err2011, snms, wm, "mu")
b <- bias_statsw_list(gti_agg11, awgts, err2011, snms, wma, "mua")
bacc2011 <- rbind(a, b)
save(bacc2011, file = fp(p_data, "cropland-werr-2011.rda"))

# check error stats - do they match alternate approaches?
for(i in c("f1", "f25", "f10")) {
  for(j in c("sa30", "modmu", "glc")) {
    print(paste("cross-checking calculations in", i, j))
    a1 <- bias_statsw(gti_agg11[[i]], awgts[[i]], err2011[[i]], snms, wm,
                      "mu", aweight = FALSE)[, j, with = FALSE]
    a2 <- bias_statsw(gti_agg11[[i]], awgts[[i]], err2011[[i]], snms, wm,
                      "mu", rweight = FALSE, aweight = FALSE)[,j, with=FALSE]
    a3 <- bias_statsw(gti_agg11[[i]], awgts[[i]], err2011[[i]], snms, wm,
                      "mu")[, j, with = FALSE]
    c1 <- getValues(err2011[[i]][[j]])
    w1 <- getValues(gti_agg11[[i]])
    w2 <- getValues(awgts[[i]])
    print("non-area weighted mean matches?")
    print(a1 == round(weighted.mean(c1, w1, na.rm = TRUE), 2))
    print("totally unweighted mean matches?")
    print(a2 == round(cellStats(err2011[[i]][[j]], mean), 2))
    print("double weighted mean matches?")
    print(a3 == round(weighted.mean(c1, w1 * w2, na.rm = TRUE), 2))
  }
}

# Subset difference grids, for 2007
snms <- c("sa30", "globmu", "modmu", "glc")
err2007 <- lapply(lev, function(x) {
  sapply(dlist_act[snms], function(y) list(y[[x]]$g2007))
})
names(err2007) <- lev
gti_agg07 <- lapply(gti_agg, function(x) x$g2007)

a <- bias_statsw_list(gti_agg07, awgts, err2007, snms, wm, "mu")
b <- bias_statsw_list(gti_agg07, awgts, err2007, snms, wma, "mua")
bacc2007 <- rbind(a, b)

fact <- c(1, 5, 10, 25, 50, 100)
alph <- c(225, 40)
x <- c(0, 3, 6, 9, 12, 15)
w <- 3 / 8
xo <- (cumsum(rep(w, 8)) - w / 2)[-c(2, 4, 6, 8)]
o <- c(0, w)
xa <- sapply(x, function(x) x + xo)
cx <- c(1.25, 1, 1)
g1 <- "grey90"
reg <- c("Country", "Agricultural", "Density independent")
lcnms <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
mutype <- c("mua", "mu")
# svec <- paste(gsub("f", "", bacc2011[stnm == "mu", ol]), "km")
svec <- bacc2011[stnm == "mu", ol]
xl <- c(-0.5, 18)
yl <- c(-30, 50)
shd <- c(4.5, 10.5, 16.5)
cols <- c("red", "orange3", "green4", "blue")
colnms <- snms
pchs <- c("+", "o", "x")
yax <- seq(yl[1], yl[2], 5)
xax <- seq(1, 6, 5 / (length(yax) - 1))

pdf(fp(p_fig, "fig1b.pdf"), height = 7, width = 7)
par(mar = rep(1, 4), oma = c(2, 2, 0, 0), mgp = c(1, 0.5, 0), tcl = -0.3)
plot(xl, yl, pch = "", yaxt = "n", xaxt = "n", xaxs = "i", yaxs = "i",
     ylab = "", xlab = "")#,
for(i in shd) polyfunc2(x = i, y = yl, w = 3, col = g1, bcol = g1, lwd = 1)
abline(h = yax, v = NULL, col = "grey80", lty = 1)
polyfunc2(x = 8.75, y = yl, w = 18.5, col = "transparent", bcol = "black")
lines(c(-1, 18), c(0, 0), lwd = 2, col = "grey80")
# i <- 1; j <- 1; k <- 1
for(i in 1:length(svec)) {
  lv <- svec[i]
  for(j in 1:length(snms)) {
```

```r
    # pchs1 <- pchs
    nm <- snms[j]
    for(k in 1:length(mutype)) {
      dat <- bacc2011[ol == lv & stnm == mutype[k], nm, with = FALSE][[1]]
      pcol <- makeTransparent(cols[j], alpha = alph[k])
      polyfunc2(xa[j, i] + o[k], c(0, dat), w = w, col = pcol, bcol = pcol)
    }
  }
}
mgp <- c(2, 0.2, 0)
axis(1, at = seq(1.5, 18.5, 3), labels = fact, tcl = 0.4, mgp = mgp)
axis(2, at = yax, labels = yax, las = 2, tcl = 0.4, mgp = mgp)
mtext(side = 1, text = "Resolution (km)", outer = TRUE, line = 0.5)
mtext(side = 2, text = "Bias/MAE (%)", outer = TRUE, line = 1)
legend(x = 12.4, y = -19.9, legend = lcnms, pch = 15, col = cols, adj = 0,
       pt.cex = 1.5, bty = "n", cex = 0.8, x.intersp = 0.5)
legend(x = 11.9, y = -19.9, legend = rep("", 4), pch = 15, adj = 0,
       col = makeTransparent(cols, alpha = alph[2]), pt.cex = 1.5, bty = "n",
       cex = 0.8)
text(x = 12.7, y = -20.5, labels = "MAE", srt = 45, adj = c(0, 0), cex= 0.8)
text(x = 12.2, y = -20.5, labels = "Bias", srt = 45, adj = c(0, 0), cex = 0.8)
dev.off()

# Difference between 2007 and 2011 bias and accuracy statistics
baccdiff <- bacc2011[, snms, with = FALSE] -  bacc2007[, snms, with = FALSE]
setnames(baccdiff, colnames(baccdiff), mcap)

# baccdiff <- round(rbind(baccdiff, baccdiff[, lapply(.SD, mean)]), 2)
baccdiff <- cbind("Resolution" = rep(paste(fact, "km"), 2), baccdiff)
xtable::xtable(baccdiff)
```

5.3.2 Other calculations methods

Across country, agricultural region only, or bin-wise (cropland density classes) mean.

```r
load(spathfunc("lcf_out2011.rda"))
lcf <- data.table(lcf_out11)

fact <- c(1, 5, 10, 25, 50, 100)
alph <- c(225, 40)
x <- c(0, 3, 6, 9, 12, 15)
w <- 3 / 8
xo <- (cumsum(rep(w, 8)) - w / 2)[-c(2, 4, 6, 8)]
o <- c(0, w)
xa <- sapply(x, function(x) x + xo)
cx <- c(1.25, 1, 1)
g1 <- "grey90"
reg <- c("Country", "Agricultural", "Density independent")
lcnms <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
mutype <- c("MAE", "Bias")
svec <- colnames(lcf)[grep("km", colnames(lcf))]
xl <- c(-0.5, 18)
yl <- c(-40, 40)
```

```r
shd <- c(4.5, 10.5, 16.5)
cols <- c("red", "orange3", "green4", "blue")
colnms <- names(lcf)[4:length(names(lcf))]
pchs <- c("+", "o", "x")
yax <- seq(yl[1], yl[2], 5)
xax <- seq(1, 6, 5 / (length(yax) - 1))

pdf(fp(p_fig, "cropland_bias_region.pdf"), height = 7, width = 7)
par(mar = rep(1, 4), oma = c(2, 2, 0, 0), mgp = c(1, 0.5, 0), tcl = -0.3)
plot(xl, yl, pch = "", yaxt = "n", xaxt = "n", xaxs = "i", yaxs = "i",
     ylab = "", xlab = "")#,
for(i in shd) polyfunc2(x = i, y = yl, w = 3, col = g1, bcol = g1, lwd = 1)
abline(h = yax, v = NULL, col = "grey80", lty = 1)
polyfunc2(x = 8.75, y = yl, w = 18.5, col = "transparent", bcol = "black")
lines(c(-1, 18), c(0, 0), lwd = 2, col = "grey80")
# i <- 1; j <- 1; k <- 1
for(i in 1:length(svec)) {
  lv <- colnms[i]
  for(j in 1:length(snms)) {
    pchs1 <- pchs
    nm <- lcnms[j]
    for(k in 1:length(mutype)) {
      #dat <- get(mutype[k])[V1 == snms[j] & V2 %in% LC2, get(lev_vec[i])]
      dat <- lcf[Metric == mutype[k] & Map == nm, lv, with = FALSE]
      pcol <- makeTransparent(cols[j], alpha = alph[k])
      polyfunc2(xa[j, i] + o[k], range(dat), w = w, col = pcol, bcol = pcol)
      for(z in 1:length(reg)) {
        lpfunc(x[i] + xo[j] + o[k], dat[z], col = "black", size = cx[z],
               pch = pchs[z], type = "pt")
      }
    }
  }
}

axis(1, at = seq(1.5, 18.5, 3), labels = fact)
axis(2, at = yax, labels = yax, las = 2)
mtext(side = 1, text = "Resolution (km)", outer = TRUE, line = 0.5)
mtext(side = 2, text = "Bias/MAE (%)", outer = TRUE, line = 1)
legend(x = 12.4, y = -19.9, legend = lcnms, pch = 15, col = cols, adj = 0,
       pt.cex = 1.5, bty = "n", cex = 0.8, x.intersp = 0.5)
legend(x = 11.9, y = -19.9, legend = rep("", 4), pch = 15, adj = 0,
       col = makeTransparent(cols, alpha = alph[2]), pt.cex = 1.5, bty = "n",
       cex = 0.8)
text(x = 12.7, y = -20.5, labels = "MAE", srt = 45, adj = c(0, 0), cex= 0.8)
text(x = 12.2, y = -20.5, labels = "Bias", srt = 45, adj = c(0, 0), cex = 0.8)
legend(x = 12.1, y = 40,
       legend = c("Country", "Agricultural", "Density independent"),
       pch = pchs,
       pt.cex = c(1.5, 1.5), bty = "n", cex = 0.8)
dev.off()
```

5.3.3 Bin density plots

To see how many values contribute to bin-wise means.

```r
# Using the union of agricultural areas, rather than whole country
bin_size <- do.call(rbind, lapply(lev, function(i) {
  # i <- "f50"
  st <- mubias_2011[bin != "all" & ol == i & il == "sa30" & bvals == "mu",
                    .(bin, N)]
  st[, bin := as.integer(bin)]  # convert bin to integers
  setkey(st, bin)
  st[data.table(bin = 1:20), N]  # merge to bin key to allow NAs
}))

pdf(fp(p_fig, "cropland_bins.pdf"), height = 7, width = 7)
par(mfrow = c(3, 2), mar = c(2, 2, 2, 2), oma = c(3, 3, 0, 0),
    mgp = c(1, 0.25, 0), tcl = -0.1)
for(i in 1:nrow(bin_size)) {
  barplot(bin_size[i, ], las = 2, col = "grey70", names.arg = seq(5, 100, 5))
  # axis(1, at = 1:20, labels = seq(5, 100, 5), las = 2)
  mtext(side = 3, text = paste(fact[i], "km"))
}
mtext(side = 1, text = "% Cropland", outer = TRUE)
mtext(side = 2, text = "Bin count", outer = TRUE, line = 1)
dev.off()

# check bin frequencies starting a bit higher: 2.5% cover
binv <- seq(2.5, 97.5, 5)
gti_bins <- lapply(gti_agg, function(x) {
  lapply(x, function(y) cut(y, breaks = binv, include.lowest = TRUE))
})
bins_25 <- lapply(gti_bins, function(x) freq(x$g2011))
barplot(bins_25$f1[1:19, 2])
```

6 District-level cropland density versus map accuracy

6.1 Data

```r
library(croplandbias)
library(nlme)
library(mgcv)
library(ape)
library(fitdistrplus)

# Paths
p_root <- proj_root("croplandbias")
p_fig <- fp(p_root, "croplandbias/inst/paper/figures/")
p_data <- fp(p_root, "croplandbias/inst/extdata/landcover")
# p_data2 <- fp(p_root, "croplandbias/data/")

# Load in datasets
data(sashp)  # SA shape
data(gadm)
# load(fp(p_data2, "ZAF_adm2.rda"))  # magisterial districts
load(spathfunc("d_grid_act.rda"))  # actual diffence grids
load(spathfunc("d_grid_abs.rda"))  # actual diffence grids
paths <- dir(p_data, pattern = , full.names = TRUE)
gti <- raster(spathfunc("cover2011sum_mask.tif"))  # gti 2011
crs(gti) <- crs(sashp)
namask <- raster(fp(p_data, "namask.tif"))  # NA mask
gti <- mask(gti, namask)  # apply mask to GTI

# Transform magisterial districts, remove Prince Edward Islands
md <- spTransform(gadm, CRSobj = sashp@proj4string)
md <- md[md$ID_2 != 313, ]
```

Mean area of magisterial district.

```
round(mean(rgeos::gArea(md, byid = TRUE) / 1000000))
```

```
## [1] 3445
```

6.2 Analyses

6.2.1 Extract data for magisterial districts

Both cropland percentages and absolute errors.

```
# GTI data
# mean cropland cover
gti_md <- extract(x = gti, y = md, progress = "text")
save(gti_md, file = fp(p_data, "gti_md.rda"))
# load("external/ext_data/gti_md.rda")
gti_mu <- t(sapply(gti_md, function(x) {
  c(length(x), length(which(is.na(x))), round(mean(x, na.rm = TRUE), 1))
}))

# mean no zeros
gti_mu0 <- t(sapply(gti_md, function(x) {
  c(length(x), length(which(is.na(x))), round(mean(x[x > 0], na.rm = TRUE), 1))
}))

# how many contribute when non-cropland removed, and which are non-zero
nonzero <- sapply(gti_md, function(x) length(x[!is.na(x) & (x > 0)]))
whichnonzero <- sapply(gti_md, function(x) which(!is.na(x) & (x > 0)))

# median
gti_median <- t(sapply(gti_md, function(x) {
  c(length(x), length(which(is.na(x))), round(median(x, na.rm = TRUE), 1))
}))

gti_median0 <- t(sapply(gti_md, function(x) {
  c(length(x), length(which(is.na(x))),
    round(median(x[x > 0], na.rm = TRUE), 1))
}))
```

```r
par(mar = c(0, 0, 0, 0))
plot(md[323, ])
plot(gti, add = TRUE)
plot(md[323, ], add = TRUE)

# Absolute biases at 1 km
snms <- c("sa30", "globmu", "modmu", "glc")
dlist_abs1km <- stack(lapply(dlist_abs[snms], function(x) x$f1$g2011))
# cellStats(!is.na(dlist_abs1km), sum)
crs(dlist_abs1km) <- crs(sashp)
dlist_abs1km_md <- extract(x = dlist_abs1km, y = md, progress = "text")
save(dlist_abs1km_md, file = fp(p_data, "dlist_abs1km_md.rda"))
# load(fp(p_data, "dlist_abs1km_md.rda"))

# check counts, primarily of NA values for each dataset
dlist_counts <- t(sapply(dlist_abs1km_md, function(x) {
  apply(x, 2, function(y) {
    c(length(y), length(which(is.na(y))))#, round(mean(y, na.rm = TRUE), 1))
  })
}))

chkfun <- function(x, tol = 0.001) {
  return(apply(x, 1, function(x) abs(max(x) - min(x)) < tol))
}
a <- seq(1, 7, 2)
b <- seq(2, 8, 2)
dlist_counts[which(!chkfun(dlist_counts[, a])), a]
dlist_counts[which(!chkfun(dlist_counts[, b])), b]

# bind stats
# MD
# some have more NAs than others, so select which LC set has max NAs
nas <- apply(dlist_counts[, seq(2, 8, 2)], 1, max)
all(gti_mu[, 1] == dlist_counts[, 1])

# then calculate mean per LC set per district
# mean
dlist_mu <- cbind(nas, t(sapply(dlist_abs1km_md, function(x) {
  round(colMeans(x, na.rm = TRUE), 1)
})))

# mean no zero
dlist_mu0 <- cbind(nas, t(sapply(1:length(dlist_abs1km_md), function(x) {
  tab <- round(dlist_abs1km_md[[x]])
  sel <- whichnonzero[[x]]  # select out reference cropland pixels only
  round(apply(tab, 2, function(x) mean(x, na.rm = TRUE)), 1)
})))

# plot(dlist_mu0[, 2], dlist_mu02[, 2])
# i <- 2
# plot(gti_mu0[, 3], dlist_mu02[, i])
# plot(gti_mu0[, 3], dlist_mu0[, i])
# median
dlist_median <- cbind(nas, t(sapply(dlist_abs1km_md, function(x) {
  round(matrixStats::colMedians(x, na.rm = TRUE), 1)
})))
colnames(dlist_median) <- c("nas", snms)

dlist_median0 <- cbind(nas, t(sapply(1:length(dlist_abs1km_md), function(x) {
  tab <- round(dlist_abs1km_md[[x]])
  sel <- whichnonzero[[x]]  # select out reference cropland pixels only
  round(apply(tab, 2, function(x) median(x, na.rm = TRUE)), 1)
})))


# bind function
bind_stats <- function(x, y, coords = xy,
                       snms = c("sa30", "globmu", "modmu", "glc")) {
  dat <- cbind.data.frame(x, y)
  dat$nd <- apply(dat[, c(2, 4)], 1, max)  # max of GTI and LC NAs
  colnames(dat)[c(1, 3)] <- c("N", "gti")
  dat <- dat[, -2]
  dat <- cbind(coords, dat)
  dat$dmu4 <- round(rowMeans(dat[, snms]), 1)  # mean across all 4
  dat$dmu3 <- round(rowMeans(dat[, snms[snms != "globmu"]]), 1) # mu - globcov
  dat
}

# centroids for districts
xy <- as.data.frame(rgeos::gCentroid(md, byid = TRUE))  # get centroid coords

# bind to gti for comparisons
mu <- bind_stats(x = gti_mu, y = dlist_mu)
mu$w <- mu$N - mu$nd  # weight - N contributing pixels
mu0 <- bind_stats(x = gti_mu0, y = dlist_mu0)
mu0$w <- nonzero  # weight - N contributing pixels (non-zeros)
med <- bind_stats(x = gti_median, y = dlist_median)
med$w <- med$N - med$nd  # weight - N contributing pixels
med0 <- bind_stats(x = gti_median0, y = dlist_median0)
med0$w <- nonzero  # weight - N contributing pixels (non-zeros)
```

6.2.2 Run regressions and check for spatial autocorrelation

```r
# check the polygons and rows line up for coordinates
# i <- 300
# a <- extract(x = gti, y = md[i, ], progress = "text")
# length(a[[1]])
# length(which(is.na(a[[1]])))
# mean(a[[1]], na.rm = TRUE)
# mu[i, ]

proc_df <- function(x, nm) {
  nms <- which(colnames(x) %in% c("x", "y", "gti", nm, "w"))
  dat <- x[, nms]
  colnames(dat)[3:4] <- c("pv", "dv")
  return(dat)
}

nms <- colnames(mu)[match(c(snms, "dmu4", "dmu3"), colnames(mu))]
lm_mu <- lapply(nms, function(x) {
  dat <- proc_df(x = mu, nm = x)
  dlm <- lm(dv ~ poly(pv, degree = 2), weights = w, data = dat)
  return(list("model" = dlm, "dat" = dat))
})

lm_mu0 <- lapply(nms, function(x) {
  dat <- proc_df(x = mu0, nm = x)
  dat <- dat[!is.na(dat$pv), ]
  dlm <- lm(dv ~ poly(pv, degree = 2), weights = w, data = dat)
  return(list("model" = dlm, "dat" = dat))
})

lm_med <- lapply(nms, function(x) {
  dat <- proc_df(x = med, nm = x)
  dlm <- lm(dv ~ poly(pv, degree = 2), weights = w, data = dat)
  return(list("model" = dlm, "dat" = dat))
})

lm_med0 <- lapply(nms, function(x) {
  dat <- proc_df(x = med0, nm = x)
  dat <- dat[!is.na(dat$pv), ]
  dlm <- lm(dv ~ poly(pv, degree = 2), weights = w, data = dat)
  return(list("model" = dlm, "dat" = dat))
})
```

6.2.3 Plot the fitted LM models

For evaluation only

```
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
cols <- c("red", "orange3", "green4", "blue", "black", "grey50")
lw <- c(rep(1, 4), 3, 3)
lt <- c(rep(2, 4), 1, 1)
xl <- c(0, 100)
yl <- c(0, round(max(mu[, snms] / 10 )) * 10)

# mean cropland in districts, for evaluation only
pdf(fp(p_fig, "biases_magdist.pdf"), width = 7, height = 7)
par(oma = c(5, 0, 0, 0), mar = c(5, 3, 2, 3))
plot(xl, yl, pch = "", ylab = "% bias", xlab = "% cropland", xaxs = "i",
    mgp = c(1.5, 0.5, 0), yaxs = "i")
for(j in 1:length(snms)) {
  d <- mu[, c("gti", snms[j])]
  points(d[, 1], d[, 2], pch = 20, col = cols[j], cex = 0.5)
}
for(j in 1:length(lm_mu)) {
  dlm <- lm_mu[[j]]$model
  lines(predict(dlm, newdata = data.frame("pv" = 1:110)), col = cols[j],
        lwd = lw[j], lty = lt[j])
}
legend(x = 0, y = -5, mcap, pch = 20, col = cols, bty = "n", xpd = NA,
       title = "District means")
legend(x = 80, y = -5, c(mcap, "All 4", "No GlobCover"), lty = lt, lwd = lw,
       col = cols, bty = "n", xpd = NA, title = "Fit")
dev.off()

# mean non-zero cropland
yl <- c(0, round(max(mu0[, snms] / 10, na.rm = TRUE)) * 10)
pdf(fp(p_fig, "biases_magdist_no0.pdf"), width = 7, height = 7)
par(oma = c(5, 0, 0, 0), mar = c(5, 3, 2, 3))
plot(xl, yl, pch = "", ylab = "% bias", xlab = "% cropland", xaxs = "i",
    mgp = c(1.5, 0.5, 0), yaxs = "i")
for(j in 1:length(snms)) {
  d <- mu0[, c("gti", snms[j])]
  points(d[, 1], d[, 2], pch = 20, col = cols[j], cex = 0.5)
}
for(j in 1:length(lm_mu0)) {
  dlm <- lm_mu0[[j]]$model
  lines(predict(dlm, newdata = data.frame("pv" = 1:110)), col = cols[j],
        lwd = lw[j], lty = lt[j])
}
legend(x = 0, y = -5, mcap, pch = 20, col = cols, bty = "n", xpd = NA,
       title = "District means")
legend(x = 80, y = -5, c(mcap, "All 4", "No GlobCover"), lty = lt, lwd = lw,
       col = cols, bty = "n", xpd = NA, title = "Fit")
dev.off()

# median cropland
yl <- c(0, round(max(med[, snms] / 10, na.rm = TRUE)) * 10)
pdf(fp(p_fig, "biases_magdist_med.pdf"), width = 7, height = 7)
par(oma = c(5, 0, 0, 0), mar = c(5, 3, 2, 3))
plot(xl, yl, pch = "", ylab = "% bias", xlab = "% cropland", xaxs = "i",
    mgp = c(1.5, 0.5, 0), yaxs = "i")
for(j in 1:length(snms)) {
  d <- med[, c("gti", snms[j])]
  points(d[, 1], d[, 2], pch = 20, col = cols[j], cex = 0.5)
}
for(j in 1:length(lm_med)) {
  dlm <- lm_med[[j]]$model
  lines(predict(dlm, newdata = data.frame("pv" = 1:110)), col = cols[j],
        lwd = lw[j], lty = lt[j])
}
legend(x = 0, y = -5, mcap, pch = 20, col = cols, bty = "n", xpd = NA,
       title = "District means")
legend(x = 80, y = -5, c(mcap, "All 4", "No GlobCover"), lty = lt, lwd = lw,
       col = cols, bty = "n", xpd = NA, title = "Fit")
dev.off()

# median of non-zero cropland
yl <- c(0, round(max(med0[, snms] / 10, na.rm = TRUE)) * 10)
```

```r
pdf(fp(p_fig, "biases_magdist_med0.pdf"), width = 7, height = 7)
par(oma = c(5, 0, 0, 0), mar = c(5, 3, 2, 3))
plot(xl, yl, pch = "", ylab = "% bias", xlab = "% cropland", xaxs = "i",
    mgp = c(1.5, 0.5, 0), yaxs = "i")
for(j in 1:length(snms)) {
  d <- med0[, c("gti", snms[j])]
  points(d[, 1], d[, 2], pch = 20, col = cols[j], cex = 0.5)
}
for(j in 1:length(lm_med0)) {
  dlm <- lm_med0[[j]]$model
  lines(predict(dlm, newdata = data.frame("pv" = 1:110)), col = cols[j],
        lwd = lw[j], lty = lt[j])
}
legend(x = 0, y = -5, mcap, pch = 20, col = cols, bty = "n", xpd = NA,
       title = "District means")
legend(x = 80, y = -5, c(mcap, "All 4", "No GlobCover"), lty = lt, lwd = lw,
       col = cols, bty = "n", xpd = NA, title = "Fit")
dev.off()
```

6.2.4 Check for spatial autocorrelation in residuals

```r
dists <- as.matrix(dist(mu[, c("x", "y")]))
dists_inv <- 1 / dists
diag(dists_inv) <- 0
for(i in 1:length(snms)) {
  print(Moran.I(residuals(lm_mu[[i]]$model), dists_inv)$p.value)
}

dists <- as.matrix(dist(lm_mu0[[1]]$dat[, c("x", "y")]))
dists_inv <- 1 / dists
diag(dists_inv) <- 0
for(i in 1:length(snms)) {
  print(Moran.I(residuals(lm_mu0[[i]]$model), dists_inv)$p.value)
}
```

Present and significant. To deal with it, first try to use:

6.2.5 A generalized least squares (GLS) with spatial autocovariance terms.

```r
# mean of all data
i <- "glc"  # glc
dat <- proc_df(x = mu, nm = i)
dgls <- gls(dv ~ poly(pv, 2), weights = ~1 / w, data = dat)  # orthogonal polys
dgls_exp <- update(dgls, correlation = corExp(form = ~x + y, nugget = TRUE))
plot(Variogram(dgls, form = ~x+y, resType = "n"))
plot(Variogram(dgls_exp, form = ~x+y, resType = "n"))  # funny shape remains

# cropland only mean
dat <- proc_df(x = mu0, nm = i)
dat <- dat[!is.na(dat$pv), ]
dgls <- gls(dv ~ poly(pv, 2), weights = ~1 / w, data = dat)  # orthogonal polys
```

```
dgls_exp <- update(dgls, correlation = corExp(form = ~x + y, nugget = TRUE))
plot(Variogram(dgls, form = ~x+y, resType = "n"))
plot(Variogram(dgls_exp, form = ~x+y, resType = "n"))  # funny shape remains
```

That seemed inadequate for removing autocorrelation in residuals. The GLS did not seem to properly deal with residual autocorrelation, probably because the spatial structure is complex, given the pattern of rainfall and cropland in it. A better approach is to include coordinates directly in the model, leveraging the 2-D smoothing capabilities of the GAM model in the mgcv library.

6.2.6 Generalized additive model

First check fit of model family, and whether assumptions of normally distributed errors are met.

```
# stats.stackexchange.com/questions/99425/distribution-for-percentage-data
# quartz()
# dat <- mu0[, "sa30"]
dat <- mu[, "sa30"]
dat <- dat[!is.na(dat)]
dat[dat == 0] <- dat[dat == 0] + 0.1
descdist(dat, discrete = FALSE)  # check potential distributions

# Check what distributions work best on data (non-cropland removed)
dists <- c("norm", "lnorm", "exp", "cauchy", "gamma", "logis",
           "beta","weibull")
fit_distrs <- lapply(snms, function(x) {
  dat <- mu0[, x]
  dat <- dat[!is.na(dat)] / 100
  dat[dat == 0] <- dat[dat == 0] + 0.1
  fits <- lapply(dists, function(x) fitdist(dat, x))
  names(fits) <- dists
  fits
})
names(fit_distrs) <- snms
# lapply(fit_distrs, function(x) sort(sapply(x, function(y) y$aic)))
plot(fit_distrs[[2]]$lnorm)  # lognorm shows up as best fit

# Matters more on residuals from model fit, however.
# dataset
dat <- proc_df(x = mu, nm = snms[4])  # interactively change response
# dat <- proc_df(x = med0, nm = snms[1])  # and input data
dat <- dat[!is.na(dat$pv), ]
dat[, 4] <- dat[, 4] + 0.1  # add 1 to allow for log transform

# GAM assuming Gaussian
dgam <- gam(dv ~ poly(pv, 2) + s(x, y), weights = w, data = dat,
            method = "ML")
summary(dgam)
AIC(dgam)
gam.check(dgam)
summary(fitdist(residuals(dgam), "norm"))
plot(fitdist(residuals(dgam), "norm"))

# GAM Gaussian with log link
```

```
dgam2 <- gam(dv ~ poly(pv, 2) + s(x, y), weights = w, data = dat,
            family = gaussian(link = "log"), method = "ML")
summary(dgam2)
AIC(dgam2)
gam.check(dgam2)
summary(fitdist(residuals(dgam2), "norm"))
plot(fitdist(residuals(dgam2), "norm"))

# GAM w/Gamma
dgam3 <- gam(dv ~ poly(pv, 2) + s(x, y), weights = w, data = dat,
            family = Gamma(link = "identity"), method = "ML")
summary(dgam3)
AIC(dgam3)  # bizarre large AIC
gam.check(dgam3)  # crazy deviance residuals
summary(fitdist(residuals(dgam3), "norm"))
plot(fitdist(residuals(dgam3), "norm"))

# GAM w/log
dgam4 <- gam(log(dv) ~ poly(pv, 2) + s(x, y), weights = w, data = dat,
             method = "ML")
summary(dgam4)
AIC(dgam4)
gam.check(dgam4)
summary(fitdist(residuals(dgam4), "norm"))
plot(fitdist(residuals(dgam4), "norm"))
# termplot(dgam4, terms = "poly(pv, 2)", ask = FALSE)
# pred <- predict(dgam4, newdata = newdat, type = "response", se.fit = TRUE)

# above, with smooth fit instead of quadratic
dgam4b <- gam(log(dv + 0.0001) ~ s(pv, k = 3) + s(x, y), weights = w,
              data = dat, method = "ML")
plot(dat$pv, dat$dv)
summary(dgam4b)
AIC(dgam4b)
gam.check(dgam4)
summary(fitdist(residuals(dgam4b), "norm"))
plot(fitdist(residuals(dgam4b), "norm"))

plot.gam(dgam4b, residuals = TRUE, select = 1, se = TRUE, all.terms = TRUE,
         pages = 0)
```

Testing each of the main landcover sets for model shape, the best fit in terms of AIC and assumption of normality of residuals is from a log-normal model.

6.2.7 Plot final results

Helper functions

```
predfunc <- function(model, newdat) {
  pred <- predict(model, newdata = newdat, type = "response", se.fit = TRUE)
  pdat <- transform(newdat, fit = pred$fit)
  pdat <- transform(pdat, up = fit + (1.96 * pred$se.fit),
                    lo = fit - (1.96 * pred$se.fit))
```

```r
  return(pdat)
}

termfunc <- function(model, newdat) {
  pred <- predict(model, newdata = newdat, type = "terms", se.fit = TRUE)
  pdat <- transform(newdat, fit = pred$fit[, 1] + coef(model)[1])
  pdat <- transform(pdat, up = fit + (1.96 * pred$se.fit[, 1]),
                    lo = fit - (1.96 * pred$se.fit[, 1]))
  return(pdat)
}

polyfunc <- function(pdat) {
  coord <- cbind.data.frame("x" = c(rev(pdat$pv), pdat$pv),
                            "y" = c(rev(pdat$lo), pdat$up))
}
```

6.2.7.1 GAM fits

```r
gam_mods <- lapply(nms, function(i) {
  dat1 <- proc_df(x = mu0, nm = i)
  dat1 <- dat1[!is.na(dat1$pv), ]
  dat1$dv <- dat1$dv + 0.1  # adding 0.1 to allow log transform
  dat2 <- proc_df(x = mu, nm = i)
  dat2 <- dat2[!is.na(dat2$pv), ]
  dat2$dv <- dat2$dv + 0.1  # adding 0.1 to allow log transform
  dgam1 <- gam(log(dv) ~ poly(pv, 2) + s(x, y), weights = w, data = dat1,
               method = "ML")
  dgam2 <- gam(log(dv) ~ poly(pv, 2) + s(x, y), weights = w, data = dat2,
               method = "ML")
  newdat <- with(dat1, data.frame(pv = 0:100, "x" = mean(dat1$x),
                                  "y" = mean(dat1$y)))
  pdat1 <- predfunc(dgam1, newdat)
  pdat2 <- predfunc(dgam2, newdat)
  # pdat1 <- termfunc(dgam1, newdat)
  # pdat2 <- termfunc(dgam2, newdat)
  return(list("mumod" = dgam1, "medmod" = dgam2, "pdat1" = pdat1,
              "pdat2" = pdat2,  "data1" = dat1, "data2" = dat2))
})
```

6.2.7.2 GAM plots

```r
ptcols <- sapply(cols, function(x) makeTransparent(x, 80))  # CI colors
lw <- c(rep(2, 4), 3, 3)
yl1 <- c(0, round(max(sapply(gam_mods, function(x) max(x$pdat1$fit) * 1.1)),
                  1))
yl2 <- c(0, round(max(sapply(gam_mods, function(x) max(x$pdat2$fit) * 1.1)),
                  1))
pdf(fp(p_fig, "fig2.pdf"), width = 7, height = 4.5)
par(oma = c(0, 0, 0, 0), mar = c(3, 3, 2, 2), mgp = c(1.5, 0.5, 0))
plot(xl, yl1, pch = "", ylab = "Mean Absolute Error (%)",
     xlab = "% Cropland", xaxs = "i", mgp = c(1.5, 0.5, 0), yaxs = "i",
     yaxt = "n", tcl = 0.4, mgp = c(2, 0.2, 0))
axis(2, at = 0:4, round(exp(c(-10, 1:4))), tcl = 0.4, mgp = c(2, 0.2, 0))
```

```
for(i in 1:4) {
  m <- gam_mods[[i]]
  p <- polyfunc(m$pdat1)
  polygon(p$x, p$y, col = ptcols[i], border = NA)
  lines(fit ~ pv, data = m$pdat1, col = cols[i], lwd = 1.5, lty = 1)
}
legend("topleft", mcap, lty = 1, lwd = 1.5,
       col = cols, bty = "n", xpd = NA)#, title = "Fit")
dev.off()

# for evaluation only
pdf(fp(p_fig, "biases_md_lnorm_gam_mu.pdf"), width = 7, height = 4.5)
par(oma = c(0, 0, 0, 0), mar = c(3, 3, 2, 2), mgp = c(1.5, 0.5, 0))
plot(xl, yl2, pch = "", ylab = "Mean Absolute Error (%)",
     xlab = "% Cropland", xaxs = "i", mgp = c(1.5, 0.5, 0), yaxs = "i",
     yaxt = "n")
axis(2, at = 0:4, round(exp(c(-10, 1:4))))
for(i in 1:4) {
  m <- gam_mods[[i]]
  p <- polyfunc(m$pdat2)
  polygon(p$x, p$y, col = ptcols[i], border = NA)
  lines(fit ~ pv, data = m$pdat2, col = cols[i], lwd = 1.5, lty = 1)
}
legend("topleft", mcap, lty = 1, lwd = 1.5,
       col = cols, bty = "n", xpd = NA)#, title = "Fit")
dev.off()

# Check p-values for all model terms
lapply(gam_mods, function(x) {
  m1 <- summary(x$mumod)
  m2 <- summary(x$medmod)
  rbind("mu" = round(c(m1$p.table[, 4], m1$s.pv), 3),
        "med" = round(c(m2$p.table[, 4], m2$s.pv), 3))
})  # all terms significant at p < 0.001 except 2nd order term on GlobCov med
```

6.2.7.3 Final plot of magisterial districts

For supplemental.

```
provs <- unique(md$ID_1)
greys <- paste0("grey", seq(15, 95, 10))
pdf(fp(p_fig, "md_map.pdf"), width = 7, height = 7, bg = "transparent")
par(mar = rep(0, 4))
plot(sashp, lty = 0)
for(i in 1:length(provs)) {
  plot(md[md$ID_1 == provs[i], ], col = greys[i], #border = "transparent",
       add = TRUE, lwd = 1)
}
dev.off()
```

7 Vegetative carbon bias and accuracy

Based on the Ruesch & Gibbs (2008) approach for estimating carbon density.

7.1 Data

```r
library(croplandbias)
library(RColorBrewer)
library(xtable)

# Paths
p_root <- proj_root("croplandbias")
p_fig <- fp(p_root, "croplandbias/inst/paper/figures/")
p_data <- fp(p_root, "croplandbias/inst/extdata/landcover")
p_carb <- fp(p_root, "croplandbias/inst/extdata/carbon/")
p_edat <- fp(p_root, "croplandbias/external/ext_data/")
```

7.1.1 Carbon values

Starting with the carbon look-up tables used for Africa by Ruesch & Gibbs (2008), inputs downloaded from here.

```r
cfiles <- Sys.glob(file.path(.libPaths(), "croplandbias/extdata/carbon", "m*"))
nms <- gsub("*.*m6|\\.txt", "", cfiles)
cval_list <- lapply(1:length(cfiles), function(x) {
  tab <- read.table(cfiles[x])
  tab <- tab[ c(1, 3)]
  colnames(tab) <- c("CL", nms[x])
  tab
})

# merge into single carbon table
mergefun <- function(x, y) merge(x, y, by = "CL", all.x = TRUE, all.y = TRUE)
ctab <- Reduce(mergefun, cval_list)

# Read in lookup table key
key <- readLines(spathfunc("code_key.txt", "cb"))[25:44]
key2 <- readLines(spathfunc("code_key.txt", "cb"))[47:57]
mtch <- sapply(gregexpr("[0-9]", key2), max)
lcs <- sapply(1:length(key2), function(x) substr(key2[x], 1, mtch[x]))
lcs <- gsub(" & ", ",", gsub("-", ":", lcs))

# Reshape and reduce according to carbon classes
ctab2 <- t(data.frame(sapply(lcs, function(x) {
  ind <- eval(parse(text = paste0("c(", x, ")")))
  as.numeric(as.vector(
    round(colMeans(ctab[ctab$CL %in% ind, -1], na.rm = TRUE))))
})))
rownames(ctab2) <- 1:nrow(ctab2)
colnames(ctab2) <- nms
ctab2 <- ctab2 * 0.01  # convert to tons/ha from 1000 kg/ha
```

```r
ctab2 <- cbind(transform(lcs), ctab2)
colnames(ctab2)[1] <- "class"
```

7.1.2 Further compress classes that have the same carbon value.

- 1:3, 6:8 => 1 (broadleaf and mixed forests)
- 4:5 => Drop
- 9, 10; 17 => 2 (Secondary forests, forest/cropland mosaic)
- 11, 12, 15 => 3 (shrublands)
- 20:23 => drop (water, snow, artificial surfaces)
- 19 => drop (bare areas)

```r
ctabf <- round(rbind(colMeans(ctab2[c(1, 3), 2:ncol(ctab2)], na.rm = TRUE),
                     colMeans(ctab2[10:11, 2:ncol(ctab2)], na.rm = TRUE),
                     ctab2[4:6, 2:ncol(ctab2)]))

# Read in ecofloristic regions and calculate their areas for Africa
ecoflora <- readOGR(fp(p_edat, "africa_ecofloristic_zones.sqlite"),
                    layer = "africa_ecofloristic_zones")
crs(ecoflora) <- CRS("+proj=longlat +datum=WGS84 +no_defs")
afalb <- paste0("+proj=aea +lat_1=20 +lat_2=-23 +lat_0=0 +lon_0=25 +x_0=0 ",
                "+y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs")
ecoflora_alb <- spTransform(ecoflora, CRS = CRS(afalb))
ecoflora_num <- cbind.data.frame(unique(ecoflora_alb@data),
                                 "ID" = c("06", "08", "09", "15", "NA", "18",
                                          "20", "16", "17", "WA", "19", "07"),
                                 stringsAsFactors = FALSE)
ecoflora_alb$ID <- rep("NA", nrow(ecoflora_alb))
head(ecoflora_alb)
ecoflora_alb$ID <- ecoflora_num[match(ecoflora_alb$gez_term,
                                      ecoflora_num$gez_term), "ID"]
ecoflora_alb@data[sample(1:nrow(ecoflora_alb@data), 2), ]
ecoflora_alb$area <- round(rgeos::gArea(ecoflora_alb, byid = TRUE) / 1000000,
                           1)
ecoareas <- sapply(ecoflora_num$ID, function(x) {
  sum(ecoflora_alb@data[ecoflora_alb$ID == x, "area"])
})
ecoareas <- ecoareas[!names(ecoareas) %in% c("WA", "NA")]
ecowgts <- ecoareas / sum(ecoareas)
ecowgts <- ecowgts[sort(names(ecowgts))]

i <- 1:ncol(ctabf)
ctabf$mu <- round(sapply(1:nrow(ctabf), function(x) {
  sum(ctabf[x, ] * ecowgts, na.rm = TRUE)
}), 1)
ctabf <- cbind(ctabf, t(apply(ctabf[, i], 1, function(x) range(x, na.rm=TRUE))))
colnames(ctabf)[(ncol(ctabf) - 1):ncol(ctabf)] <- c("min", "max")
rownames(ctabf) <- 1:nrow(ctabf)
LC <- c("forest", "second", "shrub", "grass", "sparse")
ctabo <- data.frame(t(ctabf[, c("mu", "min", "max")]))  # reclass table
colnames(ctabo) <- LC
```

7.2 Analysis

7.2.1 Prepare raster data

```r
load(spathfunc("d_grid_act.rda"))  # actual diffence grids
gti <- raster(spathfunc("cover2011sum_mask.tif"))  / 100  # gti 2011
namask <- raster(spathfunc("namask.tif"))  # NA mask
gti <- mask(gti, namask)  # apply mask to GTI
# cellStats(!is.na(gti), sum)

# Reconstruct original landcover estimates
snms <- c("sa30", "globmu", "modmu", "glc")
dlist_1km <- lapply(dlist_act[snms], function(x) x$f1$g2011)
lc_list <- lapply(dlist_1km, function(x) gti - x / 100)
# plot(lc_list[[1]])
# cellStats(!is.na(lc_list[[1]]), sum)
# plot(gti)

# aggregate rasters
fact <- c(5, 10, 25, 50, 100)
lc_agg <- aggregate_rast_list(fact, lc_list)   # landcover rasters
gti_agg <- aggregate_rast_list(fact, list("gti" = gti))  # GTI rasters

# need NA mask for weighting of aggregated value
namask2 <- !is.na(namask)
area_wgts <- aggregate_rast_list(fact, list(namask2), fun = sum)

# cropland cover bins, for looking at error as a function of cover
binv <- seq(0, 1, 0.05)
gti_bins <- lapply(gti_agg, function(x) {
  cut(x$gti, breaks = binv, include.lowest = TRUE)
})
```

7.2.2 Create cropland carbon estimates

```r
# Apply carbon estimates
# Function for pixel-wise carbon density from crop & non-crop fractions
carbon <- function(fcrop, cropC, noncropC) {
  carb <- fcrop * cropC + noncropC * (1 - fcrop)
  return(round(carb, 2))
}

cc <- ctab2[ctab2$class == 16, 2]
gti_c <- lapply(gti_agg, function(x) {
  r <- x$gti
  s <- stack(lapply(1:ncol(ctabo), function(y) {
    carbon(r, cc, ctabo[1, y])
  }))
  names(s) <- colnames(ctabo)
  s
})  # gti
```

```r
lc_c <- lapply(lc_agg, function(x) {
  lc <- lapply(x, function(j) {
    s <- stack(lapply(1:ncol(ctabo), function(y) {
      carbon(j, cc, ctabo[1, y])
    }))
    names(s) <- colnames(ctabo)
    s
  })
  names(lc) <- names(x)
  lc
})  # landcover datasets

# checks--right rasters being referenced?
tst <- cbind(sample(1:5, 10, replace = TRUE),
             sample(1:4, 10, replace = TRUE),
             sample(1:5, 10, replace = TRUE))
sapply(1:nrow(tst), function(i) {
  cellStats(lc_c[[tst[i, 1]]][[tst[i, 2]]][[tst[i, 3]]] -
              carbon(lc_agg[[tst[i, 1]]][[tst[i, 2]]], cc,
                     ctabo[1, tst[i, 3]]), sum)
}) # should be all zeroes

# save for further analysis
save(gti_c, lc_c, file = fp(p_carb, "carbon-tier1.rda"))
```

7.2.3 Difference the carbon datasets

```r
# percent difference
pct_diff <- function(x, y) (x - y) / x * 100
c_pct_diff <- lapply(1:length(gti_c), function(x) {
  dif <- lapply(1:length(lc_c[[x]]), function(y) {
    s <- stack(lapply(1:nlayers(lc_c[[x]][[y]]), function(z) {
      p <- pct_diff(gti_c[[x]][[z]], lc_c[[x]][[y]][[z]])
    }))
    names(s) <- colnames(ctabo)
    s
  })
  names(dif) <- names(lc_c[[x]])
  dif
})
names(c_pct_diff) <- names(lc_c)

# checks - right rasters being referenced
tst <- cbind(sample(1:5, 10, replace = TRUE),
             sample(1:4, 10, replace = TRUE),
             sample(1:5, 10, replace = TRUE))
sapply(1:nrow(tst), function(i) {
  x <- gti_c[[tst[i, 1]]][[tst[i, 3]]]
  y <- lc_c[[tst[i, 1]]][[tst[i, 2]]][[tst[i, 3]]]
  z <- c_pct_diff[[tst[i, 1]]][[tst[i, 2]]][[tst[i, 3]]]
  cellStats(z - ((x - y) / x * 100), sum)
})  # zeroes
```

```r
# Then for a map plot figure, calculate the mean pixel-wise percent difference
c_pct_diff_mu <- lapply(c_pct_diff, function(x) {
  lapply(x, function(y) calc(y, mean))
})

# # disggregate selected rasters at selected levels for plotting
# namask <- raster(fp(p_data, "namask.tif")) # load in NA mask
lev <- names(c_pct_diff_mu)
disagg <- lapply(snms, function(x) {
  l1 <- lapply(lev, function(y) {
    if(y == "f1") {
      r <- c_pct_diff_mu[[y]][[x]]
    } else {
      r <- disaggregate(c_pct_diff_mu[[y]][[x]],
                        fact = as.numeric(gsub("f", "", y)))
      r <- mask(crop(r, namask), namask)
    }
  })
  named_out(l1, lev)
})
names(disagg) <- snms

stats <- lapply(disagg, function(x) {
  sapply(x, function(y) {
    c(cellStats(y, mean), quantile(y, seq(0, 1, 0.05)))
  })
})
```

7.3 Outputs

7.3.1 Carbon error maps

```r
data(sashp)  # SA shape

# Plotting colors
lims <- c(ceiling(min(sapply(stats, function(x) x[3, ]))),
          floor(max(sapply(stats, function(x) x[21, ]))))
rng <- range(sapply(stats, range))
brks <- c(rng[1], lims[1], -45, -20, -10, -5, -1, 1, 5, 10, lims[2], rng[2])
#n_cols <- length(brks) - 1
colsall <- brewer.pal(n = 11, "Spectral")
cols <- c(colsall[1:6], "grey80", colsall[c(7, 9, 10, 11)])
legtext <- "% Difference"
cx <- 1.4
lcol <- "black"
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
lev <- names(disagg[[1]])[-c(2:3)]
lev2 <- c("1 km", "25 km", "50 km", "100 km")
pdf(fp(p_fig, "carbon_bias_map.pdf"), height = 6, width = 7)
par(mfrow = c(4, 4), mar = c(0, 0, 0, 0), oma = c(5, 5, 2, 0))
for(i in 1:length(snms)) {
  print(snms[i])
  for(j in 1:length(lev)) {
    print(lev[j])
    plot(sashp, lty = 0)
    plot(disagg[[snms[i]]][[lev[j]]], add = TRUE, col = cols,
         breaks = brks, legend = FALSE)
  if(j == 1) mtext(mcap[i], side = 2, line = 1, cex = cx)
  if(i == 1) mtext(lev2[j], side = 3, line = 0, cex = cx)
  }
}
flex_legend(ncuts = length(brks) - 1, legend.text = legtext,
            legend.vals = round(brks),
            longdims = c(0.2, 0.8), shortdims = c(0.06, 0.01),
            colvec = cols, #(length(brks) - 1),
            srt = c(270, 0), horiz = TRUE, textside = "bottom",
            legend.pos = c(4, 5), leg.adj = list(c(0.25, 0), c(0, -0.5)),
            cex.val = cx, textcol = lcol, bordercol = lcol)
dev.off()
```

7.3.2 Carbon bias/accuracy

7.3.2.1 Calculate how much country-level carbon estimates differ between datasets

```
lev_vec <- names(c_pct_diff)
# created mask for non-cropland areas, unioning GTI and each LC, filtering out
# areas of no-cropland (<1/2% total cover)
lc_union <- lapply(lev_vec, function(x) {
  lcb <- lapply(snms, function(y) {
    gti_gt0 <- Which(round(gti_agg[[x]]$gti * 100) > 0)
    lc_gt0 <- Which(round(lc_agg[[x]][[y]] * 100) > 0)
    all_gt0 <- gti_gt0 + lc_gt0
    all_gt0[all_gt0 > 0] <- 1
    all_gt0
  })
  named_out(lcb, snms)
})
names(lc_union) <- lev_vec
# plot(lc_union$f50$globmu)
# plot(gti_agg$f50$gti)

# calculate for whole country
tareas <- lapply(area_wgts, function(x) x[[1]] * 100)
gti_ctot <- sapply(1:length(gti_c), function(x) {
  cellStats(gti_c[[x]] * tareas[[x]], sum)
})  # gti total carbon stock
lc_c2 <- lapply(snms, function(x) {
  sapply(names(lc_c), function(y) lc_c[[y]][[x]])
})  # reshape lc_c2 -> landcover in outer list
names(lc_c2) <- snms
lc_ctot <- lapply(lc_c2, function(x) {
  sapply(1:length(x), function(y) {
    cellStats(x[[y]] * tareas[[y]], sum)
  })
```

```r
})  # landcover total carbon stocks
totc_cntry <- lapply(lc_ctot, function(x) {
  stats <- (gti_ctot - x) / gti_ctot * 100
  colnames(stats) <- names(lc_c2$sa30)
  rownames(stats) <- names(lc_c2$sa30$f1)
  stats
})  # country-level percent differences: gti versus landcover
names(totc_cntry) <- snms

# for cropped areas only - note here we are masking on cropland fraction only,
# not on union of gti and each landcover map, which is needed below for mean
# bias estimates
tot_area <- sum(freq(namask)[1, 2])  # sum of just the non-NA area
crop_areas <- lapply(lc_union$f1, freq)
sapply(crop_areas, function(x) x[2, 2] / tot_area)  # 29, 53, 33, 31

lc_ag2 <- lapply(snms, function(x) {
  sapply(names(lc_agg), function(y) lc_agg[[y]][[x]])
})  # reshape lc_ag2 -> landcover fractions in outer list
gti_ctot2 <- sapply(1:length(gti_c), function(x) {
  msk <- gti_agg[[x]]$gti > 0.005  # farmland > 0.05% mask
  # msk <- gti_agg[[x]]$gti > 0.05  # farmland > 0.05% mask
  msked <- raster::mask(gti_c[[x]] * tareas[[x]], msk, maskvalue = 0)
  cellStats(msked, sum)
})  # gti
lc_ctot2 <- lapply(1:length(lc_c2), function(x) {
  xx <- lc_c2[[x]]  # recycle reshaped lc_c2 list
  jj <- lc_ag2[[x]]  # recycle reshaped lc_c2 list
  sapply(1:length(xx), function(y) {
    msk <- jj[[y]] > 0.005  # farmland > 0.05% mask
    # msk <- jj[[y]] > 0  # farmland > 0.05% mask
    msked <- raster::mask(xx[[y]] * tareas[[y]], msk, maskvalue = 0)
    cellStats(msked, sum)
  })
})  # landcover carbon estimates

# but discrepancy will only be relevant at 1 km scale, because the carbon total
# keeps increasing when cropland areas are the only ones being considered
totc_crop <- lapply(lc_ctot2, function(x) {
  stats <- (gti_ctot2 - x) / gti_ctot2 * 100
  colnames(stats) <- names(lc_c2$sa30)
  rownames(stats) <- names(lc_c2$sa30$f1)
  stats
})
names(totc_crop) <- snms

# Combine tables for output to supplementals, 1 km % differences for country
# and agricultural levels
totc_out <- rbind(t(sapply(totc_cntry, function(x) x[, 1])),
                  t(sapply(totc_crop, function(x) x[, 1])))
pnms <- c("Forest", "Secondary", "Shrubland", "Grassland", "Sparse")
knms <- rep(mcap, 2)
totc_out <- cbind.data.frame(knms, round(unname(totc_out), 2))
```

```r
colnames(totc_out) <- c("Map", pnms)
totc_out <- cbind(Region = c(rep("Country", 4), rep("Agricultural", 4)),
                  totc_out)
capt <- paste("Percent differences in total carbon stock estimates",
              "calculated from the reference maps and from each of the four",
              "cropland maps. Differences are evaluated for total carbon",
              "estimates either at the country scale or over just the",
              "agricultural regions (cropland $>$0.05\\%), using",
              "the carbon densities of 5 different cover types to provide",
              "the values for the non-agricultural portions of each pixel",
              "(cover types indicated by column names).")
totc_xtab <- xtable(totc_out, caption = capt, digits = 1)
print(totc_xtab, type = "latex", caption.placement = "top",
      file = fp(p_fig, "totC-bias.tex"), include.rownames = FALSE)
```

7.3.3 Calculate bias/MAE statistics

7.3.3.1 Primary method

Weighted by **actual** cropland percentage, and then, with aggregation, by number of pixels being aggregated.

```r
# Helper functions to pass into data.table
# not used - switch on if quantiles needs
# bfn <- function(x, y) {
#   box_stats(x, weighted = TRUE, weight.opts = list("weights" = y))
# }
# bfna <- function(x, y) {
#   box_stats(abs(x), weighted = TRUE, weight.opts = list("weights" = y))
# }

wm <- function(x, w) stats::weighted.mean(x, w)
wma <- function(x, w) stats::weighted.mean(abs(x), w)

# Bias
cb_statsw_mu <- rbindlist(lapply(lev_vec, function(x) {
  il <- rbindlist(lapply(snms, function(y) {
    ref <- gti_agg[[x]][[1]]
    awgts <- area_wgts[[x]][[1]]
    rerror <- c_pct_diff[[x]][[y]]
    bstats <- bias_statsw(ref, awgts, rerror, LC, wm, "Bias", rnd = 2,
                          rweight = TRUE, aweight = TRUE, trim_wgt = TRUE)
    cbind("map" = y, bstats)
  }))
  # named_out(ol, snms)
  cbind("ol" = x, il)
}))

# calculate mean across cover types
cb_statsw_mu <- cbind(cb_statsw_mu,
                      "All" = apply(cb_statsw_mu[, LC, with = FALSE], 1, mean))

# Accuracy
cb_statsw_mua <- rbindlist(lapply(lev_vec, function(x) { # x <- "f1"
  il <- rbindlist(lapply(snms, function(y) { # y <- "sa30"
    ref <- gti_agg[[x]][[1]]
    awgts <- area_wgts[[x]][[1]]
    rerror <- c_pct_diff[[x]][[y]]
    bstats <- bias_statsw(ref, awgts, rerror, LC, wma, "MAE", rnd = 2,
                          rweight = TRUE, aweight = TRUE, trim_wgt = TRUE)
    cbind("map" = y, bstats)
  }))
  # named_out(ol, snms)
  cbind("ol" = x, il)
}))

# calculate mean across cover types
cb_statsw_mua <- cbind(cb_statsw_mua,
                       "All" = apply(cb_statsw_mua[, LC, with=FALSE], 1, mean))
# rowMeans(as.data.frame(cb_statsw_mua[1, ])[, LC])

# check error stats - do they match alternate approaches?
for(i in c("f1", "f25", "f10")) { # i <- "f1"
  for(j in c("sa30", "modmu", "glc")) { # j <- "modmu"
    for(k in c("forest", "shrub", "sparse", "second")) {  # k <- "second"
      print(paste("cross-checking calculations in", i, j, k))
      ref <- gti_agg[[i]][[1]]
      awgts <- area_wgts[[i]][[1]]
      rerror <- c_pct_diff[[i]][[j]]
      a1 <- bias_statsw(ref, awgts, rerror, LC, wm, "mu",
                        aweight = FALSE)[, get(k)]
      a2 <- bias_statsw(ref, awgts, rerror, LC, wm, "mu",
                        rweight = FALSE, aweight = FALSE)[, get(k)]
      a3 <- bias_statsw(ref, awgts, rerror, LC, wm, "mu")[, get(k)]
      c1 <- getValues(c_pct_diff[[i]][[j]][[k]])
      w1 <- getValues(gti_agg[[i]][[1]])
      w2 <- getValues(area_wgts[[i]][[1]])
      print("...non-area weighted mean matches?")
      print(a1 == round(weighted.mean(c1, w1, na.rm = TRUE), 2))
      print("...totally unweighted mean matches?")
      print(a2 == round(cellStats(rerror[[k]], mean), 2))
      print("...double weighted mean matches?")
      print(a3 == round(weighted.mean(c1, w1 * w2, na.rm = TRUE), 2))
    }
  }
}
save(cb_statsw_mua, cb_statsw_mu, file = fp(p_carb, "carbon-werr.rda"))
```

7.3.3.2 Secondary methods

```r
# selection variables
nms1 <- c(colnames(ctabo), "wgt")
nms2 <- colnames(ctabo)
# bvals <- names(box_stats(sample(1:100, 200, replace = TRUE)))
bvals <- "mu"

# check constancy of < 1/2% cropland being excluded
```

```r
areas <- sapply(lc_agg, function(x) res(x[[1]])[1]^2 / 10000)
plot(areas, (areas * 0.005))  # scales
(areas * 0.005)  # 1/2 to 5000 ha

dang <- Sys.time()
  #print(paste("..", x))
cb_stats <- lapply(lev_vec, function(x) {  # level
  #x <- lev_vec[4]
  print(paste(".", x))
  #levr <- names(c_pct_diff[[x]])
  l1 <- lapply(names(c_pct_diff[[x]]), function(y) {
    #y <- levr[1]
    print(paste("...", y))
    lc <- c_pct_diff[[x]][[y]]
    lcmask <- lc_union[[x]][[y]]

    # stack raster bins, cropland bins, and landcover set
    s <- stack(list("bin" = gti_bins[[x]], "wgt" = area_wgts[[x]][[1]],
                    "mask" = lc_union[[x]][[y]], c_pct_diff[[x]][[y]]))
    DT <- na.omit(as.data.table.raster(s))
    setkey(DT, "bin")

    # Potentially useful material deleted here: check repo prior to 17/10 if
    # needed

    fr1 <- data.table("bvals" = c("m", "m0", "ma", "ma0"))
    fr2 <- data.table("bin" = "all", "N" = nrow(DT))
    binl <- DT[, .N, by = bin]  # n obs per bin
    a <- round(rbind(DT[mask == 1, lapply(.SD, wm, wgt), .SDcols = nms1],
                     DT[, lapply(.SD, wm, wgt), .SDcols = nms1],
                     DT[mask == 1, lapply(.SD, wma, wgt), .SDcols = nms1],
                     DT[, lapply(.SD, wma, wgt), .SDcols = nms1]), 2)
    dtl <- list(DT[mask == 1, lapply(.SD, wm, wgt), by = bin, .SDcols = nms1],
                DT[, lapply(.SD, wm, wgt), by = bin, .SDcols = nms1],
                DT[mask == 1, lapply(.SD, wma, wgt), by = bin, .SDcols = nms1],
                DT[, lapply(.SD, wma, wgt), by = bin, .SDcols = nms1])
    b <- rbindlist(lapply(1:4, function(x) cbind(fr1[x], round(dtl[[x]], 2))))
    setkey(b, "bin")
    odt <- rbind(cbind(fr1, rbind(cbind(fr2, a))), binl[b])
  })
  named_out(l1, names(c_pct_diff[[x]]))
})
dut <- Sys.time() - dang  # 3.65 minutes (vs 33 in earlier incarnation),
# 24 sec for means only
names(cb_stats) <- lev_vec
#save(cb_stats, file = "external/ext_data/carbon_bias_tables.rda")
save(cb_stats, file = fp(p_carb, "carbon_bias_tables_mus.rda"))

namevec <- LC
stats <- cb_stats
i1 <- "all"
i2 <- "m"
```

```
cb_stats$f25$glc[bvals == "m" & bin != "all", lapply(.SD, mean), .SDcols = LC]
cb_stats$f25$glc[bvals == "m", ]
cb_stats$f25$glc[bvals == "m" & bin != "all", lapply(.SD, mean),
                 .SDcols = "forest"]

extract_stat0 <- function(namevec, stats, i1, i2, type = "density") {
  estats <- do.call(rbind, lapply(namevec, function(i) {
      dat <- sapply(stats, function(x) {
      vals <- unlist(sapply(x, function(y) {
        if(type == "density") {
          v <- y[bin == i1 & bvals == i2, i, with = FALSE]
        } else if(type == "nodensity") {
          v <- y[bin != i1 & bvals == i2, lapply(.SD, mean), .SDcols = i]
        }
        if(nrow(v) == 0) v <- rbindlist(list(v, as.list(NA)))
        v
      }))
    })#)
  }))
  out <- cbind(do.call(rbind, strsplit(rownames(estats), "\\.")),
               data.table(round(estats, 2)))
  #setnames(out, old = names(mu), new = "")
  return(out)
}
N <- function(namevec, stats, i1, i2) {
  estats <- data.table(do.call(rbind, lapply(namevec, function(i) {
    o <- sapply(stats, function(x) {
      sapply(x, function(y) y[bin == i1 & bvals == i2, N][1])
    })
  })))
  return(estats)
}
```

7.3.3.3 Extract statistics for secondary methods

For supplementals

```
# cropland area only
a <- extract_stat0(LC, cb_stats, "all", "m")  # mean cropland only
b <- cbind(V2 = "All", a[, lapply(.SD, mean), by = V1, .SDcols = lev_vec])
setcolorder(a, c(2:1, 3:ncol(a)))
mu <- rbind(a, b)  # bias

a <- extract_stat0(LC, cb_stats, "all", "ma")  # mean abs cropland only
b <- cbind(V2 = "All", a[, lapply(.SD, mean), by = V1, .SDcols = lev_vec])
setcolorder(b, c(2:1, 3:ncol(b)))
mua <- rbind(a, b)  # MAE

# whole country
a <- extract_stat0(LC, cb_stats, "all", "m0")  # mean whole country
b <- cbind(V2 = "All", a[, lapply(.SD, mean), by = V1, .SDcols = lev_vec])
setcolorder(b, c(2:1, 3:ncol(b)))
mu0 <- rbind(a, b)  # bias
```

```r
a <- extract_stat0(LC, cb_stats, "all", "ma0")  # mean abs whole country
b <- cbind(V2 = "All", a[, lapply(.SD, mean), by = V1, .SDcols = lev_vec])
setcolorder(b, c(2:1, 3:ncol(b)))
mu0a <- rbind(a, b)  # MAE

# density independent
a <- extract_stat0(LC, cb_stats, "all", "m0", type = "nodensity")
b <- cbind(V2 = "All", a[, lapply(.SD, mean), by = V1, .SDcols = lev_vec])
setcolorder(a, c(2:1, 3:ncol(a)))
mund <- rbind(a, b)  # bias

# check
all(round(sapply(lev_vec, function(x) {
  mean(cb_stats[[x]]$modmu[bvals == "m0" & bin != "all", forest])
}), 2) == mund[V2 == "forest" & V1 == "modmu", lev_vec, with = FALSE])

a <- extract_stat0(LC, cb_stats, "all", "ma0", type = "nodensity")
b <- cbind(V2 = "All", a[, lapply(.SD, mean), by = V1, .SDcols = lev_vec])
setcolorder(b, c(2:1, 3:ncol(b)))
munda <- rbind(a, b)  # MAE

# check
all(round(sapply(lev_vec, function(x) {
  mean(cb_stats[[x]]$modmu[bvals == "ma0" & bin != "all", forest])
}), 2) == munda[V2 == "forest" & V1 == "modmu", lev_vec, with = FALSE])
```

7.3.4 Plot mean carbon bias/MAE against scale

7.3.4.1 Density-weighted bias and accuracy

```r
cols <- c("red", "orange3", "green4", "blue")
lcnms <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
pnms <- c("Forest", "Secondary", "Shrubland", "Grassland", "Sparse")
lw <- 1.5

# yl <- range(cb_statsw_mu[, LC, with = FALSE],
#             cb_statsw_mua[, LC, with = FALSE])
# yl <- round(yl / 10) * 10
yl <- c(-150, 150)
yax <- seq(yl[1], yl[2], 10)
xax <- seq(1, 6, 5 / (length(yax) - 1))

alph <- c(225, 40)
LC2 <- c(LC[-grep("second|grass", LC)], "All")
x <- c(0, 3, 6, 9, 12, 15)
w <- 3 / 8
xo <- (cumsum(rep(w, 8)) - w / 2)[-c(2, 4, 6, 8)]
xa <- sapply(x, function(x) x + xo)
mutype <- c("cb_statsw_mua", "cb_statsw_mu")

o <- c(0, w)
pchs <- c("*", "+", "o", "-")
cx <- c(2, 1.25, 1, 1)
```

```r
g1 <- "grey90"

# ctabo
xl <- c(-0.5, 18)
shd <- c(4.5, 10.5, 16.5)

pdf(fp(p_fig, "fig3.pdf"), height = 7, width = 7)
par(mar = rep(1, 4), oma = c(2, 2, 0, 0), mgp = c(1, 0.5, 0), tcl = -0.3)
plot(xl, yl, pch = "", yaxt = "n", xaxt = "n", xaxs = "i", yaxs = "i",
     ylab = "", xlab = "")#,
for(i in shd) polyfunc2(x = i, y = yl, w = 3, col = g1, bcol = g1, lwd = 1)
abline(h = yax, v = NULL, col = "grey80", lty = 1)
polyfunc2(x = 8.75, y = yl, w = 18.5, col = "transparent", bcol = "black")
lines(c(-1, 18), c(0, 0), lwd = 2, col = "grey50")
for(i in 1:length(lev_vec)) { # i <- 1; j <- 1; k <- 1
  for(j in 1:length(snms)) {
    for(k in 1:length(mutype)) {
      dat <- get(mutype[k])[ol == lev_vec[i] & map == snms[j], LC2, with=FALSE]
      pcol <- makeTransparent(cols[j], alpha = alph[k])
      polyfunc2(xa[j, i] + o[k], range(dat), w = w, col = pcol, bcol = pcol)
      for(z in 1:length(LC2)) {
        lpfunc(x[i] + xo[j] + o[k], dat[, z, with = FALSE], col = "black",
               size = cx[z], pch = pchs[z], type = "pt")
      }
    }
  }
}

# annotate values that go beyond the range of plot (GlobCover, MODIS, 1km for.)
adj <- c(1, 0.3)
y <- 150
cx2 <- 0.8
v <- get(mutype[1])[map == "globmu" & ol == "f1", round(forest)]  # glob MAE
text(x = 1, y = y, labels = v, srt = 90, col = "grey40", adj = adj,
     cex = cx2)
adj <- c(1, 0.3)
v <- get(mutype[1])[map == "modmu" & ol == "f1", round(forest)]  # mod MAE
text(x = 1.75, y = y, labels = v, srt = 90, col = "grey40", adj = adj,
     cex = cx2)
adj <- c(0, -0.2)
v <- get(mutype[2])[map == "globmu" & ol == "f1", round(forest)]  # glob bias
text(x = 1.5, y = -y, labels = v, srt = 90, col = "grey40", adj = adj,
     cex = cx2)

# axes etc
mgp <- c(2, 0.2, 0)
axis(1, at = seq(1.5, 18.5, 3), labels = c(1, fact), tcl = 0.4, mgp = mgp)
axis(2, at = yax, labels = yax, las = 2, tcl = 0.4, mgp = mgp)
mtext(side = 1, text = "Resolution (km)", outer = TRUE, line = 0.5)
mtext(side = 2, text = "Bias/MAE (%)", outer = TRUE, line = 1)
legend(x = 12.4, y = -110, legend = lcnms, pch = 15, col = cols, adj = 0,
       pt.cex = 1.5, bty = "n", cex = 0.8, x.intersp = 0.5)
legend(x = 11.9, y = -110, legend = rep("", 4), pch = 15, adj = 0,
       col = makeTransparent(cols, alpha = alph[2]), pt.cex = 1.5, bty = "n",
       cex = 0.8)
text(x = 12.7, y = -115, labels = "MAE", srt = 45, adj = c(0, 0), cex = 0.8)
text(x = 12.2, y = -115, labels = "Bias", srt = 45, adj = c(0, 0), cex = 0.8)
legend(x = 12.1, y = 120, legend = c("Forest", "Shrubland", "Sparse", "Mean"),
       pch = pchs, pt.cex = c(2, 1.5, 1.5, 2), bty = "n", cex = 0.8)
dev.off()
```

7.3.5 Supplementary bias/accuracy tables

```r
# Statistics from cropland area-weighted measures, reshaped
cb_stats_s <- lapply(rev(mutype), function(i) {
  i0 <- do.call(rbind, lapply(snms, function(x) {
    i1 <- do.call(rbind, lapply(c(LC, "All"), function(y) {
      i2 <- do.call(cbind, lapply(lev_vec, function(z) {
        get(i)[ol == z & map == x, get(y)]
      }))
      cbind.data.frame(y, i2)
    }))
    i1 <- cbind.data.frame(x, i1)
    colnames(i1) <- c("Map", "Cover", lev_vec)
    data.table(i1)
  }))
})
names(cb_stats_s) <- c("Bias", "Accuracy")
muws <- copy(rbind(cbind(Metric = "Bias", cb_stats_s$Bias),
                   cbind(Metric = "MAE", cb_stats_s$Accuracy)))
setnames(muws, lev_vec, paste(c(1, fact), "km"))
setkeyv(muws, c("Map", "Cover"))
for(i in 1:length(LC)) muws[Cover == LC[i], Cover := pnms[i]]
setkey(muws, "Map")
for(i in 1:length(snms)) muws[Map == snms[i], Map := mcap[i]]


# Statistics from non-croplands areas included in means, for supplementals
# mu0[, c(lev_vec) := lapply(.SD, round), .SDcols = lev_vec]
# mu0a[, c(lev_vec) := lapply(.SD, round), .SDcols = lev_vec]
mu0s <- copy(rbind(cbind(Metric = "Bias", mu0),
                   cbind(Metric = "MAE", mu0a)))
setnames(mu0s, c("V1", "V2", lev_vec),
         c("Map", "Cover", paste(c(1, fact), "km")))
setkeyv(mu0s, c("Map", "Cover"))
for(i in 1:length(LC)) mu0s[Cover == LC[i], Cover := pnms[i]]
setkey(mu0s, "Map")
for(i in 1:length(snms)) mu0s[Map == snms[i], Map := mcap[i]]

# Statistics from agricultural areas
mus <- copy(rbind(cbind(Metric = "Bias", mu),
                  cbind(Metric = "MAE", mua)))
setnames(mus, c("V1", "V2", lev_vec),
         c("Map", "Cover", paste(c(1, fact), "km")))
setkeyv(mus, c("Map", "Cover"))
```

```r
for(i in 1:length(LC)) mus[Cover == LC[i], Cover := pnms[i]]
setkey(mus, "Map")
for(i in 1:length(snms)) mus[Map == snms[i], Map := mcap[i]]

# Bind all three tables together
fulls <- muws
# fulls <- rbind(cbind("Region" = "Density", muws),
#                 cbind("Region" = "Country", mu0s),
#                 cbind("Region" = "Agricultural", mus))


caption <- paste("Biases and mean absolute errors, weighted by reference",
                 "cropland density, for each of the test",
                 "maps across aggregation scales and each possible landcover",
                 "type sharing the pixel with cropland.")
# fulls_xtab <- xtable(fulls[order(Region, Metric, Cover)], digits = 1,
fulls_xtab <- xtable(fulls[order(Metric, Cover)], digits = 1,
                     caption = caption)
print(fulls_xtab, type = "latex", file = fp(p_fig, "C-bias-accuracy.tex"),
      tabular.environment = "longtable", floating = FALSE,
      caption.placement = "top", include.rownames = FALSE)
#       add.to.row = list(pos = list(0),
#                         command = "\\hline \\endhead"))
```

8 Error, bias, and accuracy in evapotranspiration estimates

Bias in PET estimates as calculated using different version of the cropland datasets to determine the vegetation properties in VIC.

8.1 Prepare datasets

Bring in cropland datasets, ET grids, reproject the latter to SA Albers, mask, convert to monthly total ET, etc.

```r
library(croplandbias)
library(RColorBrewer)

# Paths
p_root <- proj_root("croplandbias")
p_fig <- fp(p_root, "croplandbias/inst/paper/figures/")
p_data <- fp(p_root, "croplandbias/inst/extdata/landcover")
p_et <- fp(p_root, "croplandbias/inst/extdata/et")

# cropland data
data(sashp)  # SA shape
load(spathfunc("d_grid_act.rda"))  # actual diffence grids
gti <- raster(spathfunc("cover2011sum_mask.tif"))  / 100  # gti 2011
namask <- raster(spathfunc("namask.tif"))  # NA mask
gti <- mask(gti, namask)  # apply mask to GTI
# cellStats(!is.na(gti), sum)
# Reconstruct original landcover estimates
snms <- c("sa30", "globmu", "modmu", "glc")
anms <- c("gti", snms)
dlist_1km <- lapply(dlist_act[snms], function(x) x$f1$g2011)
lc_list <- lapply(dlist_1km, function(x) gti - x / 100)
# cellStats(!is.na(lc_list$sa30), sum)

# aggregate rasters
fact <- 25
lc_agg <- aggregate_rast_list(fact, lc_list)   # landcover rasters
gti_agg <- aggregate_rast_list(fact, list("gti" = gti))  # GTI rasters

# Create namask to remove all NA areas across datasets
namask2 <- !is.na(namask)  # set NAs to zero
awgts <- aggregate_rast_list(fact, list(namask2), fun = sum)
awgts <- lapply(awgts$f25, function(x) {
  r <- x[[1]]
  projection(r) <- sashp@proj4string
  r
})  # reshape a bit into list for subsequent use
names(awgts) <- "f25"
mask25k <- awgts$f25 > 0  # mask for et datasets

# read in ET estimates, but drop first year because of VIC problem in January
etf <- Sys.glob(file.path(.libPaths(), "croplandbias", "extdata", "et", "*.nc"))
etb <- lapply(etf, function(x) brick(x))
etb <- lapply(etb, function(x) dropLayer(x, i = 1:12))

# reproject them to Albers
etba <- lapply(etb, function(x) {
  b <- projectRaster(x, awgts$f25)
})

# rename layers and convert to total month mm
dts <- seq.Date(as.Date("1981/1/1"), as.Date("2008/12/31"), "days")
allmos <- substr(dts, 6, 7)
allyrs <- substr(dts, 1, 4)
ndays <- unlist(lapply(unique(allyrs), function(x) {
  unname(sapply(unique(allmos), function(y) {
    length(which(allyrs == x & allmos == y))
  }))
}))  # n days in each month in time series

mos <- format(ISOdatetime(2000,1:12,1,0,0,0),"%b")
lnames <- sapply(1981:2008, function(x) paste0(mos, "_", substr(x, 3, 4)))
lnames <- lnames[1:length(lnames)]
# length(lnames) == nlayers(etba[[1]])
etba <- lapply(etba, function(x) {
  b <- mask(x, mask25k, maskvalue = 0)
  names(b) <- lnames
  b
})
names(etba) <- anms
```

```r
# check that mm/month multiplies through correctly
tst <- etba[[1]] * ndays
plot(tst[[15]] - etba[[1]][[15]] * ndays[15])

# convert to monthly total mm
etba <- lapply(etba, function(x) x * ndays)
# plot(etba[[1]][[1]])

# date indices
yrs <- gsub("*.*_", "", lnames)
mos <- gsub("_.*", "", lnames)
yi <- t(sapply(unique(yrs), function(x) range(which(yrs == x))))  # year index
mi <- sapply(unique(mos), function(x) which(mos == x))  # month index
```

8.2 Process ET time series

Figure out which months produce maximum ET values, calculate monthly means, overall country-wide time series, etc.

```r
# Calculate mean, median, mode of dates in time series when PET max occurs
et_max <- lapply(etba, function(x) {
  bi <- stack(lapply(1:nrow(yi), function(y) {
    ind <- yi[y, 1]:yi[y, 2]
    which.max(x[[ind]])
  }))
  s <- stack(calc(bi, max), calc(bi, modal), calc(bi, mean), calc(bi, median))
  names(s) <- c("maxmax", "mode", "mean", "median")
  s
})

# plot to see which statistic picks out PET peak date most effectively
plot(et_max$gti)  # all the same, not coherent
plot(et_max$gti %in% 6:9)  # median looks most sensible and coherent
plot(et_max$gti - et_max$sa30)  # some differences in date of max
plot(et_max$gti - et_max$globmu)
plot(et_max$gti - et_max$modmu)
plot(et_max$gti - et_max$glc)

# Monthly mean values
et_momu <- lapply(etba, function(x) {
  bi <- stack(lapply(1:ncol(mi), function(y) {
    ind <- mi[, y]
    calc(x[[ind]], mean)
  }))
  names(bi) <- unique(mos)
  bi
})

# do a focal max to separate out noisy areas
et_medsm <- focal(et_max$gti$median, w = matrix(1, 3, 3), fun = modal, pad = 3)
buf <- (!is.na(et_max$gti$median) - !is.na(et_medsm)) * et_max$gti$median
et_medsm[is.na(et_medsm)] <- 0
```

```r
et_medsm <- buf + et_medsm  # fill back in areas edged out by moving window
# plot((et_medsm > 0) - (et_momu$gti[[1]] > 0))

# 1 month before and after peak
et_meds_m1 <- et_medsm - 1
# plot(et_medsm - 1)  # a few zero areas
et_meds_m1[et_meds_m1 < 1] <- 12  # set to 12 (December) if median - 1 = 0
et_meds_p1 <- et_medsm + 1
# plot(et_medsm + 1)  # nothing greater than 12

# calculate 3 month peak ET
gs_peak <- lapply(et_momu, function(x) {
  et1 <- stackSelect(x, et_meds_m1)
  et2 <- stackSelect(x, et_medsm)
  et3 <- stackSelect(x, et_meds_p1)
  stack(et1, et2, et3)
})

# Annual mean values
et_amu <- lapply(etba, function(x) {
  bi <- stack(lapply(1:nrow(yi), function(y) {
    ind <- yi[y, 1]:yi[y, 2]
    calc(x[[ind]], sum)
  }))
  et_ts <- cellStats(bi, mean)
  names(et_ts) <- 1981:2008
  list("gridmu" = calc(bi, mean), "cntry_ts" = et_ts)
})

# plot annual mean values
rbcol <- rainbow(4)
plot(1981:2008, et_amu$gti$cntry_ts, type = "l", las = 2)
for(i in 2:5) lines(1981:2008, et_amu[[i]]$cntry_ts, col = rbcol[i])

# figure out min_max years
mm_yr <- sapply(list(which.min, which.max), function(x) x(et_amu$gti$cntry_ts))

# annual totals
et_mutot <- lapply(etba, function(x) {
  bi <- stack(lapply(1:nrow(yi), function(y) {
    ind <- yi[y, 1]:yi[y, 2]
    calc(x[[ind]], sum)
  }))
})
et_mm <- lapply(et_mutot, function(x) x[[mm_yr]])
```

8.3 Calculate ET differences

```r
# difference rasters and convert to percent
# between time series annual mean
ilist <- lapply(et_amu[2:5], function(x) x[[1]])
a <- et_amu$gti$gridmu
```

```r
amu_diff <- lapply(ilist, function(b) {
  (a - b) / a * 100
})

# between time series mins
ilist <- lapply(et_mm[2:5], function(x) x[[1]])
a <- et_mm$gti[[1]]
mumin_diff <- lapply(ilist, function(b) {
  (a - b) / a * 100
})

# between time series maxs
ilist <- lapply(et_mm[2:5], function(x) x[[2]])
a <- et_mm$gti[[2]]
mumax_diff <- lapply(ilist, function(b) {
  (a - b) / a * 100
})

# between mean peak growing season ET (3 month total)
ilist <- lapply(gs_peak[2:5], function(x) calc(x, sum))
a <- calc(gs_peak$gti, sum)
gs3_diff <- lapply(ilist, function(b) {
  (a - b) / a * 100
})

# between median max growing season ET (3 month total)
ilist <- lapply(gs_peak[2:5], function(x) x[[2]])
a <- gs_peak$gti[[2]]
gsmx_diff <- lapply(ilist, function(b) {
  bl <- (a - b) / a * 100
})
```

8.4 Plot difference maps

8.4.1 Reshape and disaggregate lists

```r
# function to reshape lists for use in disaggregate_rast_list
disaggl <- function(rl, lev = "f25") {
  bo <- lapply(rl, function(x) {
    b <- list(x)
    names(b) <- lev
    b
  })
  bo
}

# reshape in list
# l2dag <- list(amu_diff, mumax_diff, mumin_diff, gs3_diff, gsmx_diff)
# names(l2dag) <- c("mu", "mumx", "mumn", "gs", "gspk")
l2dag <- list(amu_diff, mumax_diff, mumin_diff, gs3_diff)
names(l2dag) <- c("mu", "mumx", "mumn", "gs")
l2dag <- lapply(l2dag, disaggl)
```

```r
# disaggregate
l2dagd <- lapply(l2dag, function(x) {
  disaggregate_rast_list(snms, "f25", x, namask)
})

# stats for plotting
dstats <- function(dlist) {
  statsd <- lapply(dlist, function(x) {
    sapply(x, function(y) {
      c(cellStats(y, mean), quantile(y, seq(0, 1, 0.05)))
    })
  })
}
```

8.4.2 Plot maps

```r
# calculate single set of breakranges for all sets
statsl <- lapply(l2dag, dstats)
rng <- do.call(cbind, lapply(statsl, function(x) {
  ilims <- c(ceiling(min(sapply(x, function(y) y[3, ]))),
             floor(max(sapply(x, function(y) y[21, ]))))
  olims <- range(sapply(x, function(y) range(y)))
  c(olims[1], ilims, olims[2])
}))
rngdt <- data.table(t(rng))
rng <- unname(unlist(rngdt[, lapply(.SD, function(x) x[which.max(abs(x))])]))

fnm <- "et_bias_map4.pdf"
legtext <- rep("% Difference", 5)
cx <- 1.4
lcol <- "black"
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
# ecap <- c("Annual mean", "TS max", "TS min", "Peak 3", "Peak")
ecap <- c("Annual mean", "TS max", "TS min", "Peak")

rnder <- function(x) {
  dig <- nchar(max(round(x)))
  rmat <- cbind(1:8, c(1, 5, 10, 20, 50, 100, 100, 1000))
  rnd <- rmat[dig, 2]
  round(c(floor(x[1] / rnd), ceiling(x[2] / rnd))) * rnd
}

# For Supplementals
# brks <- c(seq(floor(rng[1]), -5, by = 5), -1, 1, seq(5, ceiling(rng[4]), 5))
brks <- c(-25, -15, -10, -5, -3, -1, 1, 3, 5, 10, 15, 25)
# cvec <- c("red", "orange", "grey80", "green", "blue4")
# cols <- colorRampPalette(cvec)(length(brks) - 1)
# cols[c(1, length(cols))] <- c("darkred", "purple3")
cols <- brewer.pal(length(brks) - 1, name = "RdBu")
cols[6] <- "grey80"
pdf(fp(p_fig, fnm), height = 6, width = 7)
par(mfcol = c(4, 4), mar = c(0, 0, 0, 0), oma = c(5, 5, 2, 0))
```

```r
for(j in 1:length(l2dagd)) {
  print(names(l2dagd)[j])
  for(k in 1:length(snms)) {
    rl <- l2dagd[[j]][[k]]$f25
    print(paste("...", snms[k]))
    plot(sashp, lty = 0)
    image(rl, add = TRUE, col = cols, breaks = brks, legend = FALSE)
    if(j == 1) mtext(mcap[k], side = 2, line = 1, cex = cx)
    if(k == 1) mtext(ecap[j], side = 3, line = 0, cex = cx)
  }
}
flex_legend(ncuts = length(brks) - 1, legend.text = legtext[i],
            legend.vals = brks, longdims = c(0.2, 0.8),
            shortdims = c(0.06, 0.01),
            colvec = cols, #(length(brks) - 1),
            srt = c(270, 0), horiz = TRUE, textside = "bottom",
            legend.pos = c(4, 4), leg.adj = list(c(0.1, 0.3), c(-0.25, -0.5)),
            cex.val = cx - 0.4, textcol = lcol, bordercol = lcol)
dev.off()

pdf(fp(p_fig, "fig4.pdf"), height = 6, width = 7)
par(mfcol = c(2, 2), mar = c(0, 0, 0, 0), oma = c(4, 0, 1, 0))
for(j in 1) {
  print(names(l2dagd)[j])
  for(k in 1:length(snms)) {
    rl <- l2dagd[[j]][[k]]$f25
    print(paste("...", snms[k]))
    plot(sashp, lty = 0)
    image(rl, add = TRUE, col = cols, breaks = brks, legend = FALSE)
    mtext(mcap[k], side = 3, line = -1, cex = cx)
  }
}
flex_legend(ncuts = length(brks) - 1, legend.text = legtext[i],
            legend.vals = brks, longdims = c(0.2, 0.8),
            shortdims = c(0.07, 0.01),
            colvec = cols, #(length(brks) - 1),
            srt = c(270, 0), horiz = TRUE, textside = "bottom",
            legend.pos = c(4, 4), leg.adj = list(c(0.3, 0.3), c(-0.25, -0.75)),
            cex.val = cx - 0.4, textcol = lcol, bordercol = lcol)
dev.off()
```

8.5 Calculate bias/accuracy statistics

8.5.1 Primary method (density dependent)

```r
# weighted mean functions
wm <- function(x, w) stats::weighted.mean(x, w)
wma <- function(x, w) stats::weighted.mean(abs(x), w)
snms <- c("sa30", "globmu", "modmu", "glc")

# Quick look at density dependent
# reshape for ET mu
```

```r
err2011 <- lapply("f25", function(x) {
  sapply(l2dag$mu, function(y) list(y[[x]]))
})
names(err2011) <- "f25"

# calculate bias/MAE for data
a <- bias_statsw(gti_agg$f25, awgts, err2011$f25, snms, wm, "mu")
b <- bias_statsw(gti_agg$f25, awgts, err2011$f25, snms, wma, "mua")
et_muw <- rbind(a, b)

# reshape for ET mu
err2011 <- lapply("f25", function(x) {
  sapply(l2dag$gs, function(y) list(y[[x]]))
})
names(err2011) <- "f25"

# calculate bias/MAE for data
a <- bias_statsw(gti_agg$f25, awgts, err2011$f25, snms, wm, "mu")
b <- bias_statsw(gti_agg$f25, awgts, err2011$f25, snms, wma, "mua")
et_gsw <- rbind(a, b)
```

8.5.2 Agricultural area method

```r
lev_vec <- "f25"
# create mask for non-cropland areas, unioning GTI and each LC, filtering out
# areas of no-cropland (<1/2% total cover)
lcu <- lapply(lev_vec, function(x) {
  lcb <- lapply(snms, function(y) {
    gti_gt0 <- Which(round(gti_agg[[x]]$gti * 100) > 0)
    lc_gt0 <- Which(round(lc_agg[[x]][[y]] * 100) > 0)
    all_gt0 <- gti_gt0 + lc_gt0
    all_gt0[all_gt0 > 0] <- 1
    all_gt0
  })
  named_out(lcb, snms)
})
names(lcu) <- lev_vec

# cropland cover bins, for looking at error as a function of cover
binv <- seq(0, 1, 0.05)
bins <- lapply(gti_agg[[2]], function(x) {
  cut(x[[1]], breaks = binv, include.lowest = TRUE)
})
names(bins) <- lev_vec
# bins <- cut(gti_agg$f25$gti, breaks = binv, include.lowest = TRUE)

# calculate bias stats
vnms <- names(l2dag)
bstats <- lapply(vnms, function(x) {
  a <- bias_stats_list(bins, awgts, lcu, l2dag[[x]], wm, "mu", "bias", TRUE)
  b <- bias_stats_list(bins, awgts, lcu, l2dag[[x]], wma, "mua", "bias", TRUE)
  stat <- rbind(a, b)
})
names(bstats) <- vnms

# checks to see if stats correct
# bias_stats* functions correct
chk <- bias_stats(bins$f25, awgts$f25, lcu$f25$globmu, l2dag$mu$globmu$f25, wm,
                  "mu", "bias", TRUE)
all(chk[bvals == "mu", bias] ==
      bstats$mu[il == "globmu" & bvals == "mu", bias])

# compared to raster
for(i in snms) {
  msk <- lcu$f25[[i]]
  msk[msk == 0] <- NA
  a <- l2dag$mu[[i]]$f25 * msk
  b <- awgts$f25 * msk
  d <- weighted.mean(values(a), values(b), na.rm = TRUE)
  print(round(d, 2) == bstats$mu[il == i & bvals == "mu" & bin == "all", bias])
}  # check

# extract stats
mu <- lapply(bstats, function(x) {
  extract_stat(lev_vec, snms, "all", "mu", "bias", x)
})
mua <- lapply(bstats, function(x) {
  extract_stat(lev_vec, snms, "all", "mua", "bias", x)
})

# reshape
mus <- rbindlist(lapply(names(mu), function(x) cbind(x, mu[[x]])))
mus[, ol := NULL]
setnames(mus, c("x", "il", "bias"), c("Variable", "Map", "Bias"))
setkeyv(mus, c("Map", "Variable"))
muas <- rbindlist(lapply(names(mua), function(x) cbind(x, mua[[x]])))
muas[, ol := NULL]
setnames(muas, c("x", "il", "bias"), c("Variable", "Map", "MAE"))
setkeyv(muas, c("Map", "Variable"))
mutab <- mus[muas]
mutab[, lapply(.SD, function(x) max(abs(x))), .SDcols = 3:4]
setkey(mutab, "Variable")
vnmr <- c("Annual Mean", "29-year Max", "29-year Min", "Peak")
for(i in 1:length(vnms)) mutab[Variable == vnms[i], Variable := vnmr[i]]
setkey(mutab, "Map")
for(i in 1:length(snms)) mutab[Map == snms[i], Map := mcap[i]]
mutab[order(-rank(Map))]

caption <- paste("Biases and mean absolute errors (as \\%) for",
                 "evapotranspiration variables derived from a 29-year time",
                 "series calculated by the VIC model, including the average",
                 "total ET for the",
                 "3 months of the year when ET is highest, the annual mean",
                 "and the minimum and maximum annual ETs in the time series.")
mutab_xtab <- xtable::xtable(mutab, digits = 1, caption = caption)
```

```
print(mutab_xtab, type = "latex", file = fp(p_fig, "et-bias.tex"),
      tabular.environment = "longtable", floating = FALSE,
      caption.placement = "top", include.rownames = FALSE)

et_err <- mutab
save(et_err, file = fp(p_et, "et-err-bias.rda"))
```

The largest (in an absolute sense) mean actual bias is -0.6%, while the largest mean absolute bias is 1.25%, so not much here.

9 Yield and production bias/accuracy

Based on Ramankutty et al's (2008) procedure for calculating cropland area, followed by Monfreda et al's (2008) to disaggregate yields and harvested areas.

9.1 Data

Use SA provinces with GTI to create the amount of agricultural land estimates per province, akin to provincial statistics used by Ramankutty et al (2008). Note: they mentioned SA has 11 administrative units they used, but the publications they cited here has just 9 provinces, so maybe they mistakenly counted Lesotho and Swaziland?

```
library(croplandbias)
library(xtable)
library(RColorBrewer)
# library(spatstat)

p_root <- fp(proj_root("croplandbias"), "croplandbias")
p_fig <- fp(p_root, "inst/paper/figures/")
p_edat <- fp(p_root, "external/ext_data/")
p_data <- fp(p_root, "inst/extdata/yieldprod")
#p_carb <- fp(p_root, "SAcropland/external/ext_data/carbon/")
# load(fp(p_data, "yield-bias.rda"))

# SA provinces
data(sashp)
data(prov)
prov <- prov[prov$id_1 != 9, ]  # remove Princeton Edward Islands
prov[prov$id_1 == 10, "id_1"] <- 9 # reset WC to id 10

for(i in c("d_grid_act.rda", "d_grid_act.rda")) load(spathfunc(i))
gti <- raster(spathfunc("cover2011sum_mask.tif")) / 100  # gti 2011
namask <- raster(spathfunc("namask.tif"))  # NA mask
gti <- mask(gti, namask)  # apply mask to GTI
# cellStats(!is.na(gti), sum)

# Reconstruct original landcover estimates
snms <- c("sa30", "globmu", "modmu", "glc")
dlist_1km <- lapply(dlist_act[snms], function(x) x$f1$g2011)
lc_list <- lapply(dlist_1km, function(x) gti - x / 100)
# cellStats(!is.na(lc_list$sa30), sum)
```

```r
# plot(lc_list[[1]])
# plot(gti)
# tst <- raster(spathfunc("glcsa_masked.tif"))  # check
# plot(round(tst / 100 - lc_list[[4]], 4)) # okay

# Create namask to remove all NA areas across datasets
sumna <- function(x) sum(x, na.rm = FALSE)
# namask <- calc(stack(stack(gti), stack(lclist)), sumna)
# namask[namask >= 0] <- 1
namask2 <- !is.na(namask)  # set NAs to zero
# cellStats(!is.na(gti), sum); cellStats(namask2, sum)
fact <- c(5, 10, 25, 50, 100)
```

9.2 Analyses

9.2.1 Cropland fractions

9.2.1.1 Get provincial cf factors from 1 km GTI dataset

```r
# rasterize provinces and stack with gti raster, convert to data.table
provr <- rasterize(prov, y = gti, field = "id_1")#,
                   # filename = "external/ext_data/provinces.tif")
# provr <- raster("external/ext_data/provinces.tif")
provr <- mask(provr, namask)
gtis <- stack(list("prov" = provr, "f" = gti))

# calculate fractions using raster::extract with provinces, to compare speed
# prov_est <- extract(gti, prov, progress = "text")
# prov_estNA <- lapply(prov_est, function(x) x[!is.na(x)])
# cftest <- cbind("carea" = sapply(prov_estNA, sum),
#                 "parea" = sapply(prov_estNA, length),
#                 "cf" = sapply(prov_estNA, sum) / sapply(prov_estNA, length))

# data.table calculations
gti_dt <- na.omit(as.data.table.raster(gtis, xy = TRUE))
setkey(gti_dt, prov)
cf <- gti_dt[, list("carea" = sum(f), "parea" = length(f),
                    "cf" = sum(f) / length(f)), by = prov]
# cftest - cf[, 2:4, with = FALSE]  # close, but diff caused by few NAs in provr
# data.table approach is much more efficient
```

9.2.1.2 Calculate cropland fractions from 1 km landcover sets

Derived at 1 km resolution rather 10 km (as in Ramankutty et al, 2008), using data.tables to calculate provincial-level correction factors for each

```r
lcs <- stack(list("prov" = provr, stack(lc_list)))
lcs_dt <- na.omit(as.data.table.raster(lcs, xy = TRUE))
setkey(lcs_dt, prov)
fc <- function(x) sum(x) / length(x)  # fraction functions
lc_cf <- lapply(list(sum, length, fc), function(x) {
  lcs_dt[, lapply(.SD, x), by = prov, .SDcols = snms]
})
```

```r
names(lc_cf) <- colnames(cf)[-1]
# lcs_dt[, .N, by = prov] == gti_dt[, .N, by = prov]

# correction factors
pcf <- cbind("prov" = prov$id_1,
             sapply(snms, function(x) cf$cf / lc_cf$cf[, get(x)]))
pcf <- data.table(pcf)
```

9.2.1.3 Re-scale landcover cropland fraction using cf factors

```r
# multiply each lc fraction value by correction factor for the particular
# province
lcs_dta <- copy(lcs_dt)
for(j in snms) {
  for(g in pcf$prov) {
    lcs_dta[prov == g, (j) := (get(j) * pcf[prov == g, get(j)])]
  }
}

# check to see if DT syntax correct
for(i in 1:9) {
  for(j in snms) {
    print(all(lcs_dta[prov == i, get(j)][1:10] ==
                (lcs_dt[prov == i, get(j)] * pcf[i, get(j)])[1:10]))
  }
}

# check to see if the factors resulted in same total area per province
fchk <- lcs_dta[, lapply(.SD, sum), by = prov, .SDcols = snms]
par(mfrow = c(2, 2), mar = rep(1, 4))
for(i in snms) plot(cf[, carea], fchk[, get(i)])  # yup

# how many pixels have fraction > 1
for(i in 1:9) {
  for(j in snms) {
    print(paste(j, ":", i, ":",
                lcs_dta[prov == i, length(which(get(j) > 1))] /
                  lcs_dta[prov == i, .N]))
  }
}  # several in globcover, modis, and glc

# Adjust these to equal 1
lcs_dta2 <- copy(lcs_dta)

# function to adjust fractions to fall within 1 while maintaining statistics
force1 <- function(x) {
  a <- x
  while(any(a > 1)) {
    reall <- sum(a[a > 1] - 1)  # total of parts of x > 1, to reallocate
    ind <- which(a < 1)  # parts of x less than 1
    a[a > 1] <- 1
    af <- (reall + sum(a[a < 1])) / sum(a[a < 1])
    a[a < 1] <- a[a < 1] * af
  }
  return(a)
}
# check if works on modis set
hist(lcs_dta[prov == 8, modmu])
hist(force1(lcs_dta[prov == 8, modmu]))
round(sum(force1(lcs_dta[prov == 8, modmu])), 2) ==
  round(sum(lcs_dta[prov == 8, modmu]), 2)

for(j in snms) {
  for(g in pcf$prov) {
    lcs_dta2[prov == g, (j) := force1(get(j))]
  }
}

# Check to see if all fractions <= 1 and that provincial sums are equal
for(i in 1:9) {
  for(j in snms) {
    print(paste(j, ":", i, ":",
                lcs_dta2[prov == i, length(which(get(j) > 1))] /
                  lcs_dta[prov == i, .N], ":",
                round(lcs_dta2[prov == i, sum(get(j))], 4) ==
                  round(lcs_dta[prov == i, sum(get(j))], 4)))
  }
}  # check

# convert back to rasters
lc_adj <- dt_to_raster(lcs_dta2, CRSobj = sashp@proj4string)
lc_adj <- dropLayer(lc_adj, i = 1)
```

9.2.1.4 Aggregate adjusted rasters, compare cropland fractions

```r
tl <- lapply(1:4, function(x) lc_adj[[x]])
names(tl) <- snms
lc_agg <- aggregate_rast_list(fact, tl) # landcover rasters
rm(tl)
gtic <- crop(gti, lc_adj$sa30)
gti_agg <- aggregate_rast_list(fact, list("gti" = gtic))  # GTI rasters

# set up masks and weights for assessing output bias and absolute errors
namask <- calc(stack(gtic, lc_agg$f1), sumna)
# cellStats(!is.na(gtic), sum); cellStats(!is.na(lc_agg$f1$sa30), sum)
namask[namask >= 0] <- 1
namask2 <- !is.na(namask)  # set NAs to zero
# cellStats(!is.na(namask), sum); cellStats(namask2, sum)
fact <- c(5, 10, 25, 50, 100)
awgts <- aggregate_rast_list(fact, list(namask2), fun = sum)
awgts <- lapply(awgts, function(x) x[[1]])  # unlist on inner loop

# compare extents to make sure no offsets - look at in QGIS
# writeRaster(gti_agg$f1$gti, filename = "external/ext_data/test/gti_crop.tif")
# writeRaster(lc_agg$f1$sa30, filename = "external/ext_data/test/sa30_crop.tif")
# looks fine
# tst <- gti_agg$f5$gti - lc_agg$f5$sa30
# plot(tst * 100)
#pct_diff <- function(x, y) (x - y) / x * 100

# calculate differences between adjusted fraction and reference
cf_pct_diff <- lapply(snms, function(x) {
  dif <- lapply(names(lc_agg), function(y) {
    d <- gti_agg[[y]][[1]] - lc_agg[[y]][[x]]
  })
  named_out(dif, names(lc_agg))
})
names(cf_pct_diff) <- snms

# same, but as percent
cf_pct_diff2 <- lapply(snms, function(x) {
  dif <- lapply(names(lc_agg), function(y) {
    d <- (gti_agg[[y]][[1]] - lc_agg[[y]][[x]]) * 100
  })
  named_out(dif, names(lc_agg))
})
names(cf_pct_diff2) <- snms

# cropland cover bins, for looking at error as a function of cover, based on
# original extent of gti data, for ease
binv <- seq(0, 1, 0.05)
bins <- lapply(gti_agg, function(x) {
  cut(x$gti, breaks = binv, include.lowest = TRUE)
})

lev <- names(cf_pct_diff[[1]])
lcu <- lapply(lev, function(x) {
  lcb <- lapply(snms, function(y) {
    gti_gt0 <- Which(gti_agg[[x]][[1]] > 0)
    lc_gt0 <- Which(lc_agg[[x]][[y]] > 0)
    all_gt0 <- gti_gt0 + lc_gt0
    all_gt0[all_gt0 > 0] <- 1
    all_gt0
  })
  named_out(lcb, snms)
})
names(lcu) <- lev

# calculate bias/MAE for data
# calculate means
wm <- function(x, w) stats::weighted.mean(x, w)
wma <- function(x, w) stats::weighted.mean(abs(x), w)

# reshape error raster for density-weighted bias/accuracy calculations
cf_pct_resh <- lapply(lev, function(x) {
  sapply(cf_pct_diff2, function(y) list(y[[x]]))
})
names(cf_pct_resh) <- lev
```

```r
# bias/accuracy
a <- bias_statsw_list(gti_agg, awgts, cf_pct_resh, snms, wm, "mu")
b <- bias_statsw_list(gti_agg, awgts, cf_pct_resh, snms, wma, "mua")
lcf_err <- rbind(a, b)


# check
# Older variant of code from compare-landcover.Rmd to evaluate whether newer DT
# version is finding correct results
# check error stats - do they match alternate approaches?
for(i in c("f1", "f25", "f10")) {
  for(j in c("sa30", "modmu", "glc")) {
    print(paste("cross-checking calculations in", i, j))
    a1 <- bias_statsw(gti_agg[[i]]$gti, awgts[[i]], cf_pct_resh[[i]], snms, wm,
                      "mu", aweight = FALSE)[, j, with = FALSE]
    a2 <- bias_statsw(gti_agg[[i]]$gti, awgts[[i]], cf_pct_resh[[i]], snms, wm,
                      "mu", rweight = FALSE, aweight = FALSE)[,j, with=FALSE]
    a3 <- bias_statsw(gti_agg[[i]]$gti, awgts[[i]], cf_pct_resh[[i]], snms, wm,
                      "mu")[, j, with = FALSE]
    c1 <- getValues(cf_pct_resh[[i]][[j]])
    w1 <- getValues(gti_agg[[i]]$gti)
    w2 <- getValues(awgts[[i]])
    print("non-area weighted mean matches?")
    print(a1 == round(weighted.mean(c1, w1, na.rm = TRUE), 2))
    print("totally unweighted mean matches?")
    print(a2 == round(cellStats(cf_pct_resh[[i]][[j]], mean), 2))
    print("double weighted mean matches?")
    print(a3 == round(weighted.mean(c1, w1 * w2, na.rm = TRUE), 2))
  }
}

# Removed whole-country and agricultural area accuracy measures. See repo prior
# to 18/10 if needed.
svec <- c("mu", "mua")
aa <- c("Bias", "Accuracy")
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")

# Reshape for output table
lcf_out <- do.call(rbind, lapply(1:length(svec), function(i) {
  a <- do.call(rbind.data.frame, lapply(snms, function(x) {
    aa <- sapply(lev, function(y) {
      lcf_err[ol == y & stnm == svec[i], x, with = FALSE]
    })
    named_out(c(x, aa), c("Map", paste(c(1, fact), "km")))
  }))
  named_out(cbind.data.frame(aa[i], a),
            c("Metric", "Map", paste(c(1, fact), "km")))
}))

# rename landcover sets for output
lcf_out <- as.data.table(lcf_out)
setkey(lcf_out, "Map")
for(i in 1:length(snms)) lcf_out[Map == snms[i], Map := mcap[i]]
```

```r
# Output table
caption <- paste("Bias and mean absolute errors (MAE) in statistically",
                 "constrained cropland maps across aggregation scales,",
                 "weighted by density of cropland",
                 "cover in the reference map. ")
lcfout_xtab <- xtable(lcf_out[order(Metric)], digits = 1, caption = caption)
print(lcfout_xtab, type = "latex",
      file = fp(p_fig, "cropadj-bias-accuracy.tex"),
      tabular.environment = "longtable", floating = FALSE,
      caption.placement = "top", include.rownames = FALSE)

# disaggregate for plotting
# namaskc <- crop(namask, gtic)
disagg <- disaggregate_rast_list(snms, lev, cf_pct_diff, namask)

stats <- lapply(disagg, function(x) {
  sapply(x, function(y) {
    c(cellStats(y * 100, mean), quantile(y * 100, seq(0, 1, 0.05)))
  })
})
```

9.2.1.5 Plot for supplementary data

```r
lims <- c(ceiling(min(sapply(stats, function(x) x[3, ]))),
          floor(max(sapply(stats, function(x) x[21, ]))))
rng <- range(sapply(stats, range))
brks <- c(rng[1], -50, -30, -20, -10, -5, -1, 1, 5, 10, 20, 30, 50, rng[2])
cols <- colorRampPalette(c("red", "grey80", "blue4"))(length(brks) - 1)

legtext <- "% Difference"
cx <- 1.4
lcol <- "black"
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
lev <- names(disagg[[1]])[-c(2:3)]
lev2 <- c("1 km", "25 km", "50 km", "100 km")
pdf(fp(p_fig, "cropland_adj_bias_map2.pdf"), height = 6, width = 7)
par(mfrow = c(4, 4), mar = c(0, 0, 0, 0), oma = c(5, 5, 2, 0))
for(i in 1:length(snms)) {
  print(snms[i])
  for(j in 1:length(lev)) {
    print(lev[j])
    plot(sashp, lty = 0)
    plot(disagg[[snms[i]]][[lev[j]]] * 100, add = TRUE, col = cols,
         breaks = brks, legend = FALSE)
  if(j == 1) mtext(mcap[i], side = 2, line = 1, cex = cx)
  if(i == 1) mtext(lev2[j], side = 3, line = 0, cex = cx)
  }
}
flex_legend(ncuts = length(brks) - 1, legend.text = legtext,
            legend.vals = round(brks),
            longdims = c(0.2, 0.8), shortdims = c(0.06, 0.01),
            colvec = cols, #(length(brks) - 1),
            srt = c(270, 0), horiz = TRUE, textside = "bottom",
            legend.pos = c(4, 5), leg.adj = list(c(0.25, 0), c(0, -0.5)),
            cex.val = cx, textcol = lcol, bordercol = lcol)
dev.off()
```

9.2.2 Yield disaggregation

Following Monfreda et al's (2008) methods. They appear to have used the 2002 district census, so we will use the more recent 2007 census.

9.2.2.1 Process magisterial districts and census data

```r
# Load in yield dataset and magisterial district polygons
data(gadm)  # magisterial districts
yld <- fread(spathfunc("maize_wheat_2007.csv", "yl"))

# Transform magisterial districts, remove Prince Edward Islands
md <- spTransform(gadm, CRSobj = sashp@proj4string)
md <- md[md$ID_2 != 313, ]
md@data <- md@data[, c("PID", "ID_2", "NAME_1", "NAME_2")]  # trim down columns
# fix a few names that occur more than once to make unique for merging properly
md@data[md$ID_2 == 43, "NAME_2"] <- "MiddelburgEC"
md@data[md$ID_2 == 143, "NAME_2"] <- "RichmondKZ"
md@data[md$ID_2 == 86, "NAME_2"] <- "HeidelbergG"

# mean MD area
round(mean(rgeos::gArea(md, byid = TRUE) / 1000000))  # 3445 km square

# check names in two datasets
# match(md@data$NAME_2, yld$district)
# match(yld$district, md@data$NAME_2)
# yld$district[which(!yld$district %in% md@data$NAME_2)]
# sort(md@data$NAME_2[which(!md@data$NAME_2 %in% yld$district)])

# some offline fixing of names ensues
# write.csv(md@data, file = "external/ext_data/mdrect.csv")
# writeOGR(md, dsn = "external/ext_data/mdcheck.sqlite", layer = "mdcheck",
#          driver = "SQLite")

# reread fixed data, merge a few districts' data where there are synonyms or 1
# district subsumed by another--sum yields, because these districts were
# probably broken into smaller pieces
# fix same names for merging properly
yld[district == "Middelburg" & province == "Eastern Cape",
    district := "MiddelburgEC"]
yld[district == "Richmond" & province == "KwaZulu-Natal",
    district := "RichmondKZ"]
yld[district == "Heidelberg" & province == "Gauteng",
    district := "HeidelbergG"]

# sumna <- function(x) sum(x, na.rm = TRUE)
mrg <- rbindlist(lapply(unique(yld$merge)[-1], function(x) {
  yld[merge == x, lapply(.SD, sumna), .SDcols = grep("mz|wh", names(yld))]
}))
nms <- sapply(unique(yld$merge)[-1], function(x) {
  nm <- yld[merge == x, district]
})
mnms <- unname(sapply(nms, function(x) x[x %in% md$NAME_2][1]))

keep <- names(yld)[-c(2:5)]
yld2 <- rbind(yld[!district %in% unname(unlist(nms)), keep, with = FALSE],
              cbind("district" = mnms, mrg))  # adjusted yield dataset

# merge with magisterial district data
mdyld <- sp::merge(md@data, yld2, by.x = "NAME_2", by.y = "district",
                   all.x = TRUE)  # merge
mdyld <- mdyld[match(mdyld$NAME_2, md@data$NAME_2), ]  # reorder rows correctly
nrow(mdyld) == nrow(md@data)  # check

# calculate yields and total planted ha
mdyld$mzha <- rowSums(mdyld[, grep("mz_ha", colnames(mdyld))])
mdyld$mzyld <- round(rowSums(mdyld[, grep("mz_pr", colnames(mdyld))]) /
                       mdyld$mzha, 1)
mdyld$whha <- rowSums(mdyld[, grep("wh_ha", colnames(mdyld))])
mdyld$whyld <- round(rowSums(mdyld[, grep("wh_pr", colnames(mdyld))]) /
                       mdyld$whha, 1)

# join to md data
m <- mdyld[match(md$ID_2, mdyld$ID_2),
           c("ID_2", "NAME_2", "mzha", "mzyld", "whha", "whyld")]
colnames(m)[1:2] <- c("id", "name")
md@data <- cbind(md@data, m)
all(md$NAME_2 == md$name); all(md$ID_2 == md$id)
md@data <- md@data[, -c(2:4)]

# plot to make sure districts didn't get reordered at all
# par(mar = rep(0, 4))
# plot(sashp)
# plot(md, add = TRUE)
# plot(md[md$id == 214, ], col = "red", add = TRUE)
# plot(md[md$id == 327, ], col = "red", add = TRUE)
# plot(md[md$id == 263, ], col = "red", add = TRUE)
# plot(md[md$name == "Barberton", ], col = "red", add = TRUE)
```

9.2.2.2 Calculate individual crop fractions

Rasterize MDs and calculate individual crop fractions (for maize), for both GTI and adjusted LC cropland fractions

```r
mdr <- rasterize(md, provr, field = "id")  # write to temp space if needed
mds <- stack(list("md" = crop(mdr, gti_agg$f1$gti), "gti" = gti_agg$f1$gti,
                  lc_adj))  # use gti and lc_adj
# v <- values((lc_agg$f1$modmu > 1) * 1); sum(v[!is.na(v)]); rm(v)

# data.table calculations
md_dt <- as.data.table(mds, xy = TRUE)
```

```r
setkey(md_dt, md)
mdarea <- md_dt[, .N, by = md]  # how many km2 in each MD
setnames(mdarea, "N", "mdkm2")
mdarea <- mdarea[!is.na(md)]  # remove NAs
setkey(mdarea, md)
md_dt <- na.omit(md_dt)  # remove NAs from fraction data.tables
varea <- md_dt[, .N, by = md]  # area of non-NA data in MDs
setnames(varea, "N", "vkm2")
setkey(varea, md)
mdva <- mdarea[varea][, fmd:= round(vkm2 / mdkm2, 2)]  # md valid area
# mdcf[, fmd := mdarea[varea][, round(vkm2 / mdkm2, 2)]]  # fraction non-NA in md

# yield data for MDs
ydt <- data.table(md@data[, c("id", "mzha", "mzyld", "whha", "whyld")])
setnames(ydt, "id", "md")
setkey(ydt, "md")
mdvay <- mdva[ydt][is.na(mzha), c("mzha", "mzyld") := 0]
mdvay[is.na(whha), c("whha", "whyld") := 0]
mdvay[, c("mdkm2", "vkm2") := NULL]
# mdcfy[, mfrac := round(((mzha * fmd) / 100) / carea, 4)]  # maize fraction

# calculate crop fractions for each dataset
md_dta <- copy(md_dt)
# md_dt[, lapply(.SD, mean), by = md, .SDcols = c("gti", snms)]
md_lcl <- lapply(c("gti", snms), function(j) {
  DT <- md_dta[, list("carea" = round(sum(get(j)), 2),  # crop area
                      "parea" = length(get(j)),  # non-NA area in MD
                      "cf" = round(sum(get(j)) / length(get(j)), 4)),
               by = md]
  # calculate crop fraction, adjusting ha first by how much non-NA area there
  # is in MD
  DTy <- DT[mdvay][, mfrac := round(((mzha * fmd) / 100) / carea, 4)]
  DTy[is.na(mfrac), mfrac := 0]  # set mfrac to 0 in 0 cropland areas
  # print(length(which(DTy$mfrac > 1)))  # 5, 4, 15, 9, 4
  #DTy[mfrac > 1, mfrac := 1]  # set to 1 - mostly missing data causing > 1
  DTy
})
names(md_lcl) <- c("gti", snms)
# lapply(md_lcl, function(x) x[, sum(mzha)])
#length(which(is.na(md_lcl$modmu$mfrac)))
#length(which(md_lcl$sa30$mfrac > 1))

par(mfrow = c(2, 3))
for(i in 1:5) hist(md_lcl[[i]]$mfrac)
for(i in 1:5) hist(md_lcl[[i]]$carea)
plot(md_lcl[[1]]$carea, md_lcl[[3]]$carea)
for(i in 1:5) print(length(which(md_lcl[[i]]$mfrac > 1)))
```

Note: when assigning crop fractions, previously we set all fractions > 1 to 1, but now turned this off because Monfreda et al (2010) allow for double-cropping. This is likely not correct, but we are doing it here to be consistent.

9.2.2.3 Assign back crop fraction and yield to grid

```r
# first in data.tables
asnms <- c("gti", snms)
crop_ay <- lapply(c("gti", snms), function(j) {
  #j <- snms[3]
  DTya <- copy(md_dt)[, list(x, y, md, "f" = get(j))]
  DTya <- DTya[md_lcl[[j]][, list(md, fmd, mfrac, mzyld)]]
  DTya[, fa := f * mfrac]  # this is Monfreda et al equation (pg. 10)
  #print(DTya[which(round(DTya$fa, 4) > 1)[1:4], ])  # check line
  DTya[, mzya := mzyld]  # adjusted yield variable
  DTya[fa == 0, mzya := 0] # set to 0 in pixels having no crop
  DTya
})
names(crop_ay) <- asnms
sapply(crop_ay, function(x) x[, round(sum(fa), 4), by = md][, V1])

# then back to rasters for aggregation
crop_ayr <- lapply(crop_ay, function(x) {
  dt_to_raster(x[, list(x, y, fa, mzya)], CRSobj = sashp@proj4string)
})

# redo NA mask because of some slight shifts
namask <- calc(stack(lapply(crop_ayr, function(x) x$fa)), sumna)
namask[namask >= 0] <- 1
namask2 <- !is.na(namask)  # set NAs to zero
# cellStats(!is.na(gti), sum); cellStats(namask2, sum)
awgts <- aggregate_rast_list(fact, list(namask2), fun = sum)
awgts <- lapply(awgts, function(x) x[[1]])  # unlist on inner loop
# sumna <- function(x, na.rm = na.rm) sum(x, na.rm = na.rm)
# amsk <- aggregate_rast_list(fact, list(namask2), fun = sumna)
# amsk <- lapply(amsk, function(x) x[[1]])  # unlist on inner loop
```

9.2.2.4 Aggregate yields and crop area to coarser resolutions

```r
# Yield aggregation - aggregating with weighting by crop fraction
# using a custom data.table function to allow this
# note: 30/8/2017 - had to fix dtraster::dt_aggregate because of change in
# data.table.
lev <- names(disagg[[1]])
wmfun <- parse(text = "sum((mzya * fa) / sum(fa, na.rm = TRUE), na.rm = TRUE)")
sr <- crs(sashp)
d <- c(dim(gti_agg[[1]]$gti)[1:2], xres(gti_agg[[1]]$gti))  # dimensions
cyld <- lapply(lev[-1], function(i) {
  print(i)
  f <- as.integer(gsub("f", "", i))
  il <- lapply(c("gti", snms), function(j) {
    print(paste("...", j))
    rdt <- as.data.table(crop_ayr[[j]], xy = TRUE)
    o <- dt_aggregate(rdt, d[1], d[2], f, wmfun, d[3], "yw")
  })
  named_out(il, asnms)
})
names(cyld) <- lev[-1]
```

```r
# convert back to rasters
cyldr <- lapply(cyld, function(i) lapply(i, function(j) dt_to_raster(j, sr)))
cyldf1 <- lapply("f1", function(i) {
  f1yld <- lapply(asnms, function(j) crop_ayr[[j]]$mzya)
  named_out(f1yld, asnms)
})
names(cyldf1) <- lev[1]
cyld_agg <- c(cyldf1, cyldr)
# plot(cyld_agg$f10$gti)

# # check Monfreda's yield data to see how it deals with null areas
# available from http://www.earthstat.org/data-download/
# monf <- fp(fp(p_edat, "monfreda_data/maize_HarvAreaYield_NetCDF"),
#             "maize_AreaYieldProduction.nc")
# # monfmz <- brick(monf, level = 2, varname = "maizeData")
# monfmz <- raster(monf, level = 2)
#
# plot(crop(monfmz, spTransform(sashp, monfmz@crs)), col = bpy.colors(20))
# plot(crop(monfmz, spTransform(sashp, monfmz@crs)) > 0)  # almost all SA > 0

## crop areas
carea <- lapply(1:5, function(x) crop_ayr[[x]]$fa)
names(carea) <- c("gti", snms)

# sumha <- function(x, na.rm) sum(x, na.rm)
carea_agg <- aggregate_rast_list(fact, carea) # crop area, mean aggregation
#carea_agg2 <- aggregate_rast_list(fact, carea, sum) # crop area, sum agg
# plot(carea_agg$f25$globmu)
# plot(carea_agg2$f25$globmu)

# area of each pixel at each aggregation scale
agg_res <- sapply(carea_agg, function(x) res(x$gti)[1]^2 / 10000)
tareas <- lapply(awgts, function(x) x[[1]] * 100)

# plot(tareas$f50)
# plot(round((carea_agg$f50$gti * tareas$f50) -
#              (carea_agg2$f50$gti * 100), 10))  # equiv

# Removed yat check. Refer to repo prior to 18/10 to recover
# check aggregated maize fractions
for(i in lev) {
  for(j in snms) {
    ck <- crop_ay[[j]][, sum(fa)] * 100
    d <- round((ck - cellStats(tareas[[i]] * carea_agg[[i]][[j]], sum)) / ck,
               2)
    print(d)
  }
}  # all equal
```

9.2.2.5 Crop production estimates at different aggregation scales

Including differences relative to reference versions.

```r
# Type A aggregation: multiplying aggregated yields by aggregated fractions
# with crop-fraction weighted aggregate yields
cpagg1 <- lapply(1:length(cyld_agg), function(x) {
  p <- lapply(1:length(cyld_agg[[x]]), function(y) {
    (tareas[[x]] * carea_agg[[x]][[y]]) * cyld_agg[[x]][[y]]
  })
  named_out(p, asnms)
})
names(cpagg1) <- names(agg_res)

# Removed production aggregation with 0-removed yieldsother. Refer to repo prior
# to 18/10 to recover

# Type B (formerly II) aggregation: crop production differences when aggregated
# from 1 km calculate production at 1 km
cprod <- lapply(1:length(cyldf1$f1), function(x) {
  cyldf1$f1[[x]] * (carea[[x]] * 100)
})
names(cprod) <- asnms
cpaggII <- aggregate_rast_list(fact, cprod, "sum") # aggregate production

# check production estimates for consistency
# pat <- list(cpagg1, cpagg3, cpaggII)
# ptype <- c("1", "3", "II")
pat <- list(cpagg1, cpaggII)
ptype <- c("1", "II")

# check production estimates
for(i in lev[-1]) {
  print(paste("factor", i))
  for(j in asnms) {
    print(paste("...", j))
    for(k in 1:length(ptype)) {
      print(paste("...... type", ptype[k], "agg"))
      ck <- cellStats(cprod[[j]], sum)  # weighted by fraction
      ck2 <- cellStats(cprod$gti, sum) # weighted by fraction
      v <- cellStats(pat[[k]][[i]][[j]], sum)
      d <- round((ck - v) / ck, 2)
      d2 <- round((ck2 - v) / ck2, 2)
      print(c(d, d2))
    }
  }  # all types consistent across scales
}

# sapply(cprod, function(x) cellStats(x, sum))
# sapply(carea, function(x) cellStats(x, sum))

# Mean production in producing cells at different levels of aggregation.
# No need to weight by cropland density, because production value already
# factors in area - just mask out zero production areas
mup <- sapply(lev, function(i) {
  msk <- cpagg1[[i]]$gti > 0
  msk[msk == 0] <- NA
  cpmsk <- msk * cpagg1[[i]]$gti
  cellStats(cpmsk, mean)
})

# mean yield in producing cells at different levels of aggregation, use density
# weighting here
muy <- sapply(lev, function(i) {
  # i <- lev[2]; print(i)
  s <- stack(cyld_agg[[i]]$gti, carea_agg[[i]]$gti, awgts[[i]])
  names(s) <- c("yld", "ref", "awgts")
  DT <- na.omit(as.data.table.raster(s, xy = TRUE))
  DT[, wgt := ref * awgts]  # calculate weights from ref and area weights
  DT[, weighted.mean(yld, wgt)]
})  # yield is 3.361 kg/ha at all scales

# Removed original mup and muy. Refer to repo prior to 18/10 to recover

# calculate differences between them
# check first that cpagg1 and II are equivalent, make cpaggII master
for(i in lev) {
  for(j in snms) {
    a <- (cpagg1[[i]]$gti - cpagg1[[i]][[j]]) -
      (cpaggII[[i]]$gti - cpagg1[[i]][[j]])
    print(round(cellStats(a, sum), 5))
  }
}

# Production differences
# first standardize
# std_raster <- function(x) {
#   (x - cellStats(x, min)) / diff(cellStats(x, range))
# }
# cpagg_std <- lapply(cpagg1, function(x) lapply(x, function(y) std_raster(y)))
ilist <- list(names(cpagg1[[1]])[-1], names(cpagg1), "gti")
# list1 <- lapply(cpagg1, function(x) x[1])
# list2 <- lapply(cpagg1, function(x) x[-1])
list1 <- lapply(cpagg1, function(x) x[1])
list2 <- lapply(cpagg1, function(x) x[-1])
p_diff1 <- rast_list_math(ilist, list1, list2, expr = "a - b")  # type 1 agg

# Removed original p_diff2/3. Refer to repo prior to 18/10 to recover

# yield differences
list1 <- lapply(cyld_agg, function(x) x[1])
list2 <- lapply(cyld_agg, function(x) x[-1])
y_diff1 <- rast_list_math(ilist, list1, list2, expr = "a - b")  # type 1

# check grids (output in QGIS and examine for sensible results--they are)
# for(i in lev) {
#   writeRaster(y_diff1$globmu[[i]]$gti,
#               file = fp(p_data, paste0("test/globdiff", i, ".tif")))
# }
# for(i in lev) {
```

```r
#   writeRaster(list1[[i]]$gti,
#               file = fp(p_data, paste0("test/gtiyldtst", i, ".tif")))
# }
# for(i in lev) {
#   writeRaster(list2[[i]]$globmu,
#               file = fp(p_data, paste0("test/globyld2tst", i, ".tif")))
# }

# Removed original y_diff2. Refer to repo prior to 18/10 to recover

# convert them to percent differences relative to mean pixel-wise production
# here only for map plotting.
pdiffs <- lapply(list(p_diff1), function(x) {
  l1 <- lapply(snms, function(y) {
    l2 <- lapply(lev, function(z) {
      x[[y]][[z]]$gti / mup[z] * 100
    })
    named_out(l2, lev)
  })
  named_out(l1, snms)
})
names(pdiffs) <- "pd1"

ydiffs <- lapply(list(y_diff1), function(x) {
  l1 <- lapply(snms, function(y) {
    l2 <- lapply(lev, function(z) {
      x[[y]][[z]]$gti / muy[z] * 100
    })
    named_out(l2, lev)
  })
  named_out(l1, snms)
})
names(ydiffs) <- "yd1"

# disaggregate for plotting
lev <- names(y_diff1[[1]])
p_diff1d <- disaggregate_rast_list(snms, lev, pdiffs$pd1, namask)
y_diff1d <- disaggregate_rast_list(snms, lev, ydiffs$yd1, namask)

# stats for plotting
dstats <- function(dlist) {
  statsd <- lapply(dlist, function(x) {
    sapply(x, function(y) {
      c(cellStats(y, mean), quantile(y, seq(0, 1, 0.05)))
    })
  })
}
stats_pd1 <- dstats(p_diff1d)
stats_yd1 <- dstats(y_diff1d)
```

9.2.3 Plot difference maps

```r
# a raft of variables and plotting functions
dnms <- ls()[grep("[y_|p_]diff*.*d", ls())]
stnms <- ls()[grep("^stats_", ls())]
disaggl <- lapply(dnms, function(x) get(x))
statsl <- lapply(stnms, function(x) get(x))
rng <- lapply(statsl, function(x) { # x <- statsl[[1]]
  ilims <- c(ceiling(min(sapply(x, function(y) y[3, ]))),
             floor(max(sapply(x, function(y) y[21, ]))))
  olims <- range(sapply(x, function(y) range(y)))
  c(olims[1], ilims, olims[2])
})

fnms <- c("prod_bias_map.pdf", "yld_bias_map.pdf")  # output names
legtext <- rep("% Difference", 5)
cx <- 1.4
lcol <- "black"
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")
lev <- names(p_diff1d[[1]])[-c(2:3)]
lev2 <- c("1 km", "25 km", "50 km", "100 km")

rnder <- function(x) {
  dig <- nchar(max(round(x)))
  rmat <- cbind(1:8, c(1, 5, 10, 20, 50, 100, 100, 1000))
  rnd <- rmat[dig, 2]
  round(c(floor(x[1] / rnd), ceiling(x[2] / rnd))) * rnd
}
div_breaks <- function(olim, ilim, ibrk) {
  il <- rnder(ilim)
  ol <- c(floor(olim[1]), ceiling(olim[2]))
  brks <- c(ol[1], seq(il[1], -ibrk, abs(floor(il[1]) - -ibrk) / 4),
            seq(ibrk, il[2], abs(ceiling(il[2]) - ibrk) / 4), ol[2])
  brks
}

rng2 <- lapply(rng, function(x) {
  x[2:3] <- c(-100, 100)
  x
})
brksl <- lapply(rng2, function(x) {
  c(floor(x[1:2]), c(-50, -25, -10, -5, -1, 1, 5, 10, 25, 50), ceiling(x[3:4]))
})
# brksl <- list(c(-1, -0.75, -0.5, -0.25, -0.1, -0.05, -0.01, 0.01, 0.05, 0.1,
#                  0.25, 0.5, 0.75, 1),
#                c(-10.7, -7, -5, -3, -2, -1, -0.1, 0.1, 1, 2, 3, 5, 7, 10.7))

# plots start
for(i in 1:length(statsl)) {
  # i <- 2
  print(paste("figure", i, "of", length(statsl)))
  brks <- brksl[[i]]
  cols <- c(rev(brewer.pal(length(brks) / 2, "Reds")[-1]), "grey80",
          brewer.pal(length(brks) / 2, "Blues")[-1])
  brklen <- length(brks) - 1
  pdf(fp(p_fig, fnms[i]), height = 6, width = 7)
  par(mfrow = c(4, 4), mar = c(0, 0, 0, 0), oma = c(5, 5, 2, 0))
  for(j in 1:length(snms)) {
    print(snms[j])
    for(k in 1:length(lev)) {
      rl <- disaggl[[i]]
      print(lev[k])
      plot(sashp, lty = 0)
      plot(rl[[snms[j]]][[lev[k]]], add = TRUE, col = cols,
                breaks = brks, legend = FALSE)
      if(k == 1) mtext(mcap[j], side = 2, line = 1, cex = cx)
      if(j == 1) mtext(lev2[k], side = 3, line = 0, cex = cx)
    }
  }
  flex_legend(ncuts = length(brks) - 1, legend.text = legtext[i],
              legend.vals = brks, longdims = c(0.15, 0.8),
              shortdims = c(0.07, 0.01),
              colvec = cols, #(length(brks) - 1),
              srt = c(270, 0), horiz = TRUE, textside = "bottom",
              legend.pos = c(4, 5), leg.adj = list(c(0.25, 0), c(0, -0.5)),
              cex.val = cx, textcol = lcol, bordercol = lcol)
  dev.off()
}

lev <- names(p_diff1d[[1]])  # reset levs
```

Several versions of yield aggregation were tested in this analysis, each of which has a different impact on yield and production impacts. There are three types of yield aggregation that can be done (most commented out now after initial checking, refer to earlier repo versions):

1. **Type 1**: weighted mean averaging, with the weights provided by crop fractions. This produces correct country-level average yields (<1-2% different) across scales and between datasets, provided that the averaging is weighted by the cropland fractions that were aggregated to the level from which the country-level yield estimate is being calculated
2. **Type 2**: Straight averaging, which results in diluted country-level yield estimates. Incorrect, and should be obvious that this wouldn't be done, so we haven't pursued it here.
3. **Type 3**: or zero-removed averaging, where only areas having fraction of that crop contribute to the aggregated average. As with **Type 1**, this produces correct country level estimates when weighting by cropland fraction, but it does not produce correct production estimates, probably because it inflates the value of high-yielding irrigated areas out towards the dry west, where the crop fractions are small. (see below)

There are also two ways of aggregating production estimates:

1. **Type A**: Aggregating yield and crop area separately, then multiplying. This is correct with **Type 1** yield aggregation, and producing identical production estimates to **Type II**, but not with the other two types of yield aggregation.
2. **Type B**: Calculating production at the base resolution, then aggregating by sum.

Why would someone do a yield aggregation, when you have the gridded estimates? To calculate values and compare them to a coarser resolution product, for instance, or to get a country-level average.

Potential reasons to be concerned about bias in yield estimates:

- Bias in yield gap estimates, and how much closing that could affect contribution of closing gaps to boosting production.
- Country-level statistics might cancel out bias, but if bias of one type of sign is spatially correlated with the driver of values in yield gaps, then that would bias larger-scale estimates of gap closure potential.
- These concerns apply to all resolutions.

Type 2 and 3 yield aggregation are incorrect methods, and are not retained here (but see earlier commits of code, specifically prior to 18/10/2015. The other variants were used for comparison and for initial methods checks only.

9.3 Bias/MAE statistics

9.3.1 Primary method

Density weighted, following methods developed in cropland bias portion of analysis, but here metrics are based on residuals rather than percent errors, because of infinite values (harvested area/yield disaggregation methods mean that there are gridded yields/production estimates in some areas where no reference value exists, leading to infinite values if converting to percent)

```r
# yield bias/accuracy
# reshape
yerr <- lapply(lev, function(x) {
  sapply(y_diff1, function(y) list(y[[x]]$gti))
})
names(yerr) <- lev

# carea_gti <- lapply(carea_agg, function(x) x$gti)
a <- bias_statsw_list(gti_agg, awgts, yerr, snms, wm, "mu")
b <- bias_statsw_list(gti_agg, awgts, yerr, snms, wma, "mua")
yld_bacc <- rbind(a, b)

# production bias/accuracy
# reshape
perr <- lapply(lev, function(x) {
  sapply(p_diff1, function(y) list(y[[x]]$gti))
})
names(perr) <- lev

# carea_gti <- lapply(carea_agg, function(x) x$gti)
a <- bias_statsw_list(gti_agg, awgts, perr, snms, wm, "mu")
b <- bias_statsw_list(gti_agg, awgts, perr, snms, wma, "mua")
prod_bacc <- rbind(a, b)

# check error stats - do they match alternate approach?
ref <- lapply(gti_agg, function(x) x$gti)
test_err <- "perr" #"yerr"
evst <- "wma" # "wm"
for(i in c("f1", "f25", "f100")) {
  for(j in c("sa30", "modmu", "glc")) {
    print(paste("cross-checking calculations in", i, j))
    a1 <- bias_statsw(ref[[i]], awgts[[i]], get(test_err)[[i]], snms,
                      get(evst), aweight = FALSE)[, j, with = FALSE]
    a2 <- bias_statsw(ref[[i]], awgts[[i]], get(test_err)[[i]], snms,
                       get(evst), rweight = FALSE,
                       aweight = FALSE)[, j, with = FALSE]
    a3 <- bias_statsw(ref[[i]], awgts[[i]], get(test_err)[[i]], snms,
                       get(evst))[, j, with = FALSE]
    s <- stack(get(test_err)[[i]][[j]], ref[[i]], awgts[[i]])
    sv <- getValues(s)
    sv <- sv[which(!is.na(rowSums(sv))), ]  # remove NAs
    if(evst == "wm") {
      print(a1 == round(weighted.mean(sv[, 1], sv[, 2], na.rm = TRUE), 2))
      print(a2 == round(mean(sv[, 1]), 2))
      print(a3 == round(weighted.mean(sv[, 1], sv[, 2] * sv[, 3]), 2))
    } else if(evst == "wma"){
      print(a1 == round(weighted.mean(abs(sv[, 1]), sv[, 2], na.rm = TRUE), 2))
      print(a2 == round(mean(abs(sv[, 1])), 2))
      print(a3 == round(weighted.mean(abs(sv[, 1]), sv[, 2] * sv[, 3]), 2))
    }
  }
}
```

9.3.2 Secondary method

Bias and accuracy within agricultural areas

```r
# create mask for non-cropland areas, unioning GTI and LC areas with maize
# estimates -- probably not needed if doing production estimates.
lev <- names(y_diff1[[1]])

# first check aggregation to make yield aggregation variants have common area,
# so that just one lc_union is needed
# for(i in lev) {
#   for(j in snms) {
#     print(cellStats(cyld_agg[[i]][[j]] > 0 & cyld_agg3[[i]][[j]] == 0, sum))
#     #print(cellStats(carea_agg[[i]][[j]] > 0 & cyld_agg[[i]][[j]] == 0, sum))
#   }
# }  # all zeros

snms <- names(p_diff1)
lcu <- lapply(lev, function(x) {
  lcb <- lapply(snms, function(y) {
    gti_gt0 <- Which(carea_agg[[x]][[1]] > 0)
    lc_gt0 <- Which(carea_agg[[x]][[y]] > 0)
    all_gt0 <- gti_gt0 + lc_gt0
    all_gt0[all_gt0 > 0] <- 1
    all_gt0
  })
  named_out(lcb, snms)
})
names(lcu) <- lev

# cropland cover bins, for looking at error as a function of cover
binv <- seq(0, 1, 0.05)
bins <- lapply(gti_agg, function(x) {
  cut(x$gti, breaks = binv, include.lowest = TRUE)
})

# Many commented out checks removed here. Look at pre 18/10 commits to restore

# Spatial bias stats calculation on yield and production differences, not on
# percentages
# Many stats for alternate yield/production metrics removed here.
# Look at pre 18/10 commits to restore
a <- bias_stats_list(bins, awgts, lcu, p_diff1, wm, "mu", "bias", TRUE)
b <- bias_stats_list(bins, awgts, lcu, p_diff1, wma, "mua", "bias", TRUE)
d <- bias_stats_list(bins, awgts, lcu, p_diff1, sum, "sum", "bias", FALSE)
pd1_st <- rbind(a, b, d)

a <- bias_stats_list(bins, awgts, lcu, y_diff1, wm, "mu", "bias", TRUE)
b <- bias_stats_list(bins, awgts, lcu, y_diff1, wma, "mua", "bias", TRUE)
yd1_st <- rbind(a, b)

# Output statistics
p1mu <- extract_stat(lev, snms, "all", "mu", "bias", pd1_st)  # identical to p3
p1ma <- extract_stat(lev, snms, "all", "mua", "bias", pd1_st)  # ident to p3
y1mu <- extract_stat(lev, snms, "all", "mu", "bias", yd1_st)
y1ma <- extract_stat(lev, snms, "all", "mua", "bias", yd1_st)

# Older variant of code from compare-landcover.Rmd to evaluate whether newer DT
# version is finding correct results
i <- p_diff1#y_diff1 #pdiff_3 # p_diff1
jdt <- pd1_st#yd1_st # pd3_st # pd1_st
bv <- "mu" #"mua" # bv <- "mu"
for(chk in list(c("modmu", "f25"), c("glc", "f10"), c("sa30", "f50"),
                c("globmu", "f1"), c("sa30", "f1"))) {
  x <- chk[1]
  y <- chk[2]
  print(paste(".", x, "..", y))
  rs <- lcu[[y]][[x]]
  rs[rs == 0] <- NA
  # l3 <- abs(i[[x]][[y]][[1]])
  l3 <- i[[x]][[y]][[1]]
  o <- rs * l3
  w <- awgts[[y]][[1]] * rs
  wmu <- weighted.mean(getValues(o), getValues(w), na.rm = TRUE)
  print(jdt[ol == y & il == x & bvals == bv & bin == "all", bias] ==
          round(wmu, 2))
  # print(round(wmu / mup[y]), 3)
}  # check
```

9.3.3 Bias/MAE plots

9.3.3.1 Density-weighted

```r
alph <- c(225, 60)
x <- c(0, 3, 6, 9, 12, 15)
w <- 3 / 8
xo <- (cumsum(rep(w, 8)) - w / 2)[-c(2, 4, 6, 8)]
```

```r
o <- c(0, w)
xa <- sapply(x, function(x) x + xo)
cx <- c(1.25, 1)
g1 <- "grey90"

# get stats data into shape
mulw <- list(prod_bacc[stnm == "mu"], yld_bacc[stnm == "mu"])
malw <- list(prod_bacc[stnm == "mua"], yld_bacc[stnm == "mua"])
malmulw <- list(malw, mulw)  # density-weighted stats, reordered
pyl <- c("mup", "muy")

xl <- c(-0.5, 18)
yl <- c(-30, 110)  # try it out, play with it
shd <- c(4.5, 10.5, 16.5)
cols <- c("red", "orange3", "green4", "blue")
lcnms <- c("SA LC", "GlobCover", "MODIS", "GLC-Share")
pchs <- c("+", "o")

yax <- seq(yl[1], yl[2], 10)
xax <- seq(1, 6, 5 / (length(yax) - 1))

pdf(fp(p_fig, "fig5.pdf"), height = 7, width = 7)
par(mar = rep(1, 4), oma = c(2, 2, 0, 0), mgp = c(1, 0.5, 0), tcl = -0.3)
plot(xl, yl, pch = "", yaxt = "n", xaxt = "n", xaxs = "i", yaxs = "i",
     ylab = "", xlab = "")#,
for(i in shd) polyfunc2(x = i, y = yl, w = 3, col = g1, bcol = g1, lwd = 1)
abline(h = yax, v = NULL, col = "grey80", lty = 1)
polyfunc2(x = 8.75, y = yl, w = 18.5, col = "transparent", bcol = "black")
lines(c(-1, 18), c(0, 0), lwd = 2, col = "grey80")
for(ii in 1:length(lev)) { # ii <- 1; i <- 1; k <- 1
  lv <- lev[ii]
  for(i in 1:length(snms)) {
    pchs1 <- pchs
    nm <- snms[i]
    for(k in 1:length(malmulw)) {
      mm <- malmulw[[k]]
      # fetch bias stats, convert to percentage relative to mean
      # e.g. GTI mean yield
      v <- sapply(1:length(mm), function(j) {  # j <- 1
        round(mm[[j]][ol == lv, get(nm)] / get(pyl[j])[lv] * 100, 1)
      })
      pcol <- makeTransparent(cols[i], alpha = alph[k])
      polyfunc2(xa[i, ii] + o[k], range(v), w = w, col = pcol, bcol = pcol)
      xx <- rep(x[ii] + xo[i] + o[k], length(v))
      points(xx, v, pch = pchs, cex = cx)
    }
  }
}
mgp <- c(2, 0.2, 0)
axis(1, at = seq(1.5, 18.5, 3), labels = c(1, fact), mgp = mgp, tcl = 0.4)
axis(2, at = yax, labels = yax, las = 2, mgp = mgp, tcl = 0.4)
mtext(side = 1, text = "Resolution (km)", outer = TRUE, line = 0.5)
mtext(side = 2, text = "Bias/MAE (%)", outer = TRUE, line = 1)
```

```r
legend(x = 12.4, y = -10, legend = lcnms, pch = 15, col = cols, adj = 0,
       pt.cex = 1.5, bty = "n", cex = 0.8, x.intersp = 0.5)
legend(x = 11.9, y = -10, legend = rep("", 4), pch = 15, adj = 0,
       col = makeTransparent(cols, alpha = alph[2]), pt.cex = 1.5, bty = "n",
       cex = 0.8)
text(x = 12.7, y = -12, labels = "MAE", srt = 45, adj = c(0, 0), cex= 0.8)
text(x = 12.2, y = -12, labels = "Bias", srt = 45, adj = c(0, 0), cex = 0.8)
legend(x = 12.1, y = 100, legend = c("Production", "Yield"), pch = pchs,
       pt.cex = c(1.5, 1.5), bty = "n", cex = 0.8)
dev.off()
```

9.3.3.2 Agricultural area

Supplemental

```r
alph <- c(225, 60)

mula <- c("p1mu", "y1mu")
mala <- c("p1ma", "y1ma")
malmula <- list(mala, mula)
pyl <- c("mup", "muy")


p1mu

yl <- c(-70, 80)
yax <- seq(yl[1], yl[2], 10)

pdf(fp(p_fig, "yield_prod_bias_agric.pdf"), height = 7, width = 7)
par(mar = rep(1, 4), oma = c(2, 2, 0, 0), mgp = c(1, 0.5, 0), tcl = -0.3)
plot(xl, yl, pch = "", yaxt = "n", xaxt = "n", xaxs = "i", yaxs = "i",
     ylab = "", xlab = "")
for(i in shd) polyfunc2(x = i, y = yl, w = 3, col = g1, bcol = g1, lwd = 1)
abline(h = yax, v = NULL, col = "grey80", lty = 1)
polyfunc2(x = 8.75, y = yl, w = 18.5, col = "transparent", bcol = "black")
lines(c(-1, 18), c(0, 0), lwd = 2, col = "grey80")
for(ii in 1:length(lev)) {
  lv <- lev[ii]
  for(i in 1:length(snms)) {
    pchs1 <- pchs
    nm <- snms[i]
    for(k in 1:length(malmula)) {
      mm <- malmula[[k]]
      # fetch bias stats, convert to percentage relative to mean
      # e.g. GTI mean yield
      v <- sapply(1:length(mm), function(j) {
        round(get(mm[j])[ol == lv & il == nm,  bias] / get(pyl[j])[lv] * 100,1)
      })
      pcol <- makeTransparent(cols[i], alpha = alph[k])
      polyfunc2(xa[i, ii] + o[k], range(v), w = w, col = pcol, bcol = pcol)
      xx <- rep(x[ii] + xo[i] + o[k], length(v))
      points(xx, v, pch = pchs, cex = cx)
    }
  }
}
axis(1, at = seq(1.5, 18.5, 3), labels = c(1, fact))
axis(2, at = yax, labels = yax, las = 2)
mtext(side = 1, text = "Resolution (km)", outer = TRUE, line = 0.5)
mtext(side = 2, text = "Bias/MAE (%)", outer = TRUE, line = 1)
legend(x = 12.4, y = -40, legend = lcnms, pch = 15, col = cols, adj = 0,
       pt.cex = 1.5, bty = "n", cex = 0.8, x.intersp = 0.5)
legend(x = 11.9, y = -40, legend = rep("", 4), pch = 15, adj = 0,
       col = makeTransparent(cols, alpha = alph[2]), pt.cex = 1.5, bty = "n",
       cex = 0.8)
text(x = 12.7, y = -42, labels = "MAE", srt = 45, adj = c(0, 0), cex= 0.8)
text(x = 12.2, y = -42, labels = "Bias", srt = 45, adj = c(0, 0), cex = 0.8)
legend(x = 12.1, y = 65, legend = c("Production", "Yield"), pch = pchs,
       pt.cex = c(1.5, 1.5), bty = "n", cex = 0.8)

dev.off()

lapply(1:length(lev), function(ii) {
  lv <- lev[ii]
  lapply(1:length(snms), function(i) {
    nm <- snms[i]
    lapply(1:length(malmula), function(k) {
      mm <- malmula[[k]]
      v <- sapply(1:length(mm), function(j) {
        round(get(mm[j])[ol == lv & il == nm,  bias] / get(pyl[j])[lv] * 100,1)
      })
    })
  })
})

do.call(cbind, lapply(lev, function(x) {
  rbind(cbind.data.frame("Map" = y1mu[ol == x, il],
                         "v" = round(y1mu[ol == x, bias] / muy[x] * 100, 1)),
        cbind.data.frame("Map" = p1mu[ol == x, il],
                         "v" = round(p1mu[ol == x, bias] / mup[x] * 100, 1)))
}))
```

9.3.4 Supplemental tables

```r
vnms <- c(rep("Yield", length(snms)), rep("Production", length(snms)))
aa <- c("Bias", "MAE")

# Density-weighted
out_tabw <- do.call(rbind, lapply(1:2, function(x) {  # x <- 1
  out <- do.call(cbind, lapply(1:length(lev), function(ii) {  # ii <- 1
    lv <- lev[ii]
    mm <- malmulw[2:1][[x]]
    v <- do.call(rbind, lapply(length(mm):1, function(j) {  # j <- 1
      v2 <- do.call(rbind, lapply(1:length(snms), function(i) { # i <- 1
        nm <- snms[i]
        round(mm[[j]][ol == lv, get(nm)] / get(pyl[j])[lv] * 100, 1)
      }))
      rownames(v2) <- snms
      v2
    }))
  }))
  out <- cbind.data.frame(aa[x], mcap, "Variable" = vnms, unname(out))
  colnames(out) <- c("Metric", "Map", "Variable", paste(c(1, fact), "km"))
  out
}))

# Agricultural area
out_taba <- do.call(rbind, lapply(1:2, function(x) {
  out <- do.call(cbind, lapply(1:length(lev), function(ii) {
    lv <- lev[ii]
    mm <- malmula[2:1][[x]]
    v <- do.call(rbind, lapply(length(mm):1, function(j) {
      v2 <- do.call(rbind, lapply(1:length(snms), function(i) {
        nm <- snms[i]
        round(get(mm[j])[ol == lv & il == nm,  bias] / get(pyl[j])[lv] * 100,1)
      }))
      rownames(v2) <- snms
      v2
    }))
  }))
  out <- cbind.data.frame(aa[x], mcap, "Variable" = vnms, unname(out))
  colnames(out) <- c("Metric", "Map", "Variable", paste(c(1, fact), "km"))
  out
}))

# Join them
out_tab <- rbind(cbind("Region" = "Density", out_tabw),
                 cbind("Region" = "Agricultural", out_taba))

caption <- paste("Biases and mean absolute errors (MAE) in disaggregated",
                 "maize yield and production (calculated from disaggregated",
                 "yield and harvested area estimates)",
                 "maps. Values for both density-weighted and agricultural",
                 "areas bias and accuracy are presennted.",
                 "Bias and MAE were normalized to their",
                 "respective mean values calculated from reference maps.")
out_xtab <- xtable(out_tab, digits = 1, caption = caption)
print(out_xtab, type = "latex", file = fp(p_fig, "yldprod-bias.tex"),
      tabular.environment = "longtable", floating = FALSE,
      caption.placement = "top", include.rownames = FALSE)
```

9.3.5 The impacts of spatial patterns in bias

(Supplmentary analysis not presented). Using yield gap estimates as an indicator. First we'll bring in mean rainfall as a means of calculating a spatial correlation in yield gap estimates. Using data from the Princeton Global Forcing dataset, downloaded from the Africa Flood and Drought Monitor. Note: currently this is only explored here.

```
# rainfall layer prepared for SA crop subsidy analysis
load(spathfunc("mag_dist_clim.rda", "cl"))
```

```r
pre <- calc(pre2, mean)  # 31 year mean rainfall for Africa
p4s <- projection(raster(nrow = 10, ncol = 10)) # GCS
presa <- crop(pre, spTransform(sashp, p4s))  # crop to SA
projection(presa) <- p4s
presa <- projectRaster(presa, lcu$f25$sa30)  # project to Albers
presa <- raster::mask(presa, awgts$f25, maskvalue = 0)  # mask
presa1k <- disaggregate(presa, fact = 25)

# reclassify rainfalls to nearest 100 mm, tweak to catch lower and upper bound
rclmat <- cbind(seq(50, 750, 100), seq(150, 850, 100), seq(100, 800, 100))
rclmat[1] <- 15
rclmat[8, 2] <- 875
prebin <- crop(reclassify(presa1k, rclmat), namask)  # rainfall bins

# Assume a yield gap over-estimated by 100% in a particular location, say the
# 300-400 mm isohyet
prez <- (prebin == 300) | (prebin == 400)
notprez <- (prebin < 300) | (prebin > 400)
g1 <- 0
g2 <- 1

ygap_bias <- sapply(lev, function(x) {
  if(x != "f1") {
    preagg <- aggregate(prebin, fact = as.numeric(gsub("f", "", x)),
                        fun = modal)
  } else {
    preagg <- prebin
  }
  prez <- (preagg == 300) | (preagg == 400)
  notprez <- (preagg < 300) | (preagg > 400)
  sapply(snms, function(y) {
    yg <- (g1 * cyld_agg[[x]]$gti * notprez) + (g2 * cyld_agg[[x]]$gti * prez)
    yg2 <- (g1 * cyld_agg[[x]][[y]] * notprez) + (g2*cyld_agg[[x]][[y]] * prez)
    pg <- tareas[[x]] * carea_agg[[x]]$gti * yg
    pg2 <- tareas[[x]] * carea_agg[[x]][[y]] * yg2
    cellStats(pg - pg2, sum) / cellStats(pg, sum) * 100
  })
})

# Assume a yield gap over-estimated by 100% in 400 mm isohyet
ygap_bias2 <- sapply(lev, function(x) {
  if(x != "f1") {
    preagg <- aggregate(prebin, fact = as.numeric(gsub("f", "", x)), fun=modal)
  } else {
    preagg <- prebin
  }
  prez <- preagg == 400
  notprez <- preagg != 400
  sapply(snms, function(y) {
    yg <- (g1 * cyld_agg[[x]]$gti * notprez) + (g2 * cyld_agg[[x]]$gti * prez)
    yg2 <- (g1 * cyld_agg[[x]][[y]] * notprez) + (g2 * cyld_agg[[x]][[y]]*prez)
    pg <- tareas[[x]] * carea_agg[[x]]$gti * yg
    pg2 <- tareas[[x]] * carea_agg[[x]][[y]] * yg2
    cellStats(pg - pg2, sum) / cellStats(pg, sum) * 100
  })
})

# Assume a yield gap over-estimated by 100% in 500 mm isohyet
ygap_bias3 <- sapply(lev, function(x) {
  if(x != "f1") {
    preagg <- aggregate(prebin, fact = as.numeric(gsub("f", "", x)), fun=modal)
  } else {
    preagg <- prebin
  }
  prez <- preagg == 500
  notprez <- preagg != 500
  sapply(snms, function(y) {
    yg <- (g1 * cyld_agg[[x]]$gti * notprez) + (g2 * cyld_agg[[x]]$gti * prez)
    yg2 <- (g1 * cyld_agg[[x]][[y]] * notprez) + (g2 * cyld_agg[[x]][[y]]*prez)
    pg <- tareas[[x]] * carea_agg[[x]]$gti * yg
    pg2 <- tareas[[x]] * carea_agg[[x]][[y]] * yg2
    cellStats(pg - pg2, sum) / cellStats(pg, sum) * 100
  })
})

# Print out combined results to latex table
scen <- c(rep("300-400 mm", nrow(ygap_bias)), rep("400 mm", nrow(ygap_bias)),
          rep("500 mm", nrow(ygap_bias)))
ygaps <- cbind.data.frame(scen, "sensor" = rep(snms, 3),
                          rbind(ygap_bias, ygap_bias2, ygap_bias3))
ygap_xtab <- xtable::xtable(ygaps, digits = 1)
print(ygap_xtab, type = "latex",
      file = fp(p_fig, "ygap-spat-bias.tex"))

# save(list = ls(), file = fp(p_data, "yield-bias.rda"))
save(out_tab, file = fp(p_data, "yield-prod-accbias.rda"))
save(cyld_agg, cpagg1, file = fp(p_data, "yieldprod-1km.rda"))
```

The actual impact of the spatial bias is fairly low, being largely cancelled out by biases in other direction in other areas. This illustrates the advantage of a statistically constrained approach to crop area estimates.

10 Sensitivity of agent-based model to map error

Selection of magisterial districts and associated landcover sets for use in ABM example, and analysis of bias after initial allocation of agents performed based on available farmland estimated from 100 m downscaled versions of the landcover data.

10.1 Data

```
library(croplandbias)
library(readxl)
library(rgeos)

# Paths
```

```r
p_root <- fp(proj_root("croplandbias"), "croplandbias")
p_fig <- fp(p_root, "inst/paper/figures/")
p_edat <- fp(p_root, "external/ext_data/")
p_data <- fp(p_root, "inst/extdata/abm")

# landcover data
load(spathfunc("d_grid_act.rda")) # actual diffence grids
gti <- raster(spathfunc("cover2011sum_mask.tif"))  # gti 2011
namask <- raster(spathfunc("namask.tif"))  # NA mask
gti <- mask(gti, namask)  # apply mask to GTI
# cellStats(!is.na(gti), sum)

# Reconstruct original landcover estimates
snms <- c("sa30", "globmu", "modmu", "glc")
dlist_1km <- lapply(dlist_act[snms], function(x) x$f1$g2011)
lc_list <- lapply(dlist_1km, function(x) gti - x)
# lc_list <- gti - dlist_1km
# cellStats(!is.na(lc_list$sa30), sum)
lc_list <- stack(lc_list)

# magisterial district data cropland statistics (2011, but only to look at for
# selection
load(spathfunc("gti_md.rda"))
mu <- data.frame(t(sapply(gti_md, function(x) {
  c(length(x), length(which(is.na(x))), round(mean(x, na.rm = TRUE), 1))
})))
colnames(mu) <- c("N", "na", "f")

# district shapes, filtered by N missing data, minimum fraction, and area
data(gadm)
md <- spTransform(gadm, CRSobj = crs(sashp))
md <- crop(md, sashp)
mdfilt <- mu[mu$na < 20 & mu$f > 5 & mu$N < 1500, ]
mdsel <- as.numeric(row.names(mdfilt))

# Examine, including in QGIS
# plot(md)
# plot(md[mdsel[mdsel < 314], ], col = "red", add = TRUE)  # filter out Cape ones
# writeOGR(md[mdsel[mdsel < 314], ], dsn = "external/ext_data/md_abms.sqlite",
#           layer = "md_abms", driver = "SQLite")
```

10.1.1 Magisterial District selection

Looking at the data in QGIS, including some rainfall maps from earlier papers and the inter-tubes, it looks like the following districts are a good match:

- Clocolan (269)
- Ficksburg (274)
- Fouriesburg (275)
- Marquard (288)

```r
# md <- md[mdsel[mdsel < 314], ]
abm_md <- md[mdsel[mdsel %in% c(269, 274:275, 288)], ]
```

```r
# writeOGR(abm_md, dsn = "external/ext_data/md4.sqlite", layer = "md4",
#          driver = "SQLite")

# Deprecated: earlier used when running 2007 rather than 2011
round(rgeos::gArea(abm_md, byid = TRUE) / 10000 / 100)
```

```
##  269  274  275  288
## 1190 1329 1037 1268
```

10.1.2 Extract cropland percentage from selected districts

```r
gti4 <- crop(gti, extent(abm_md))
lc4 <- crop(lc_list, extent(abm_md))
cb <- brick(stack(gti4, lc4))
cpct <- extract(gti4, abm_md)
dcrop_pct <- round(sapply(cpct, sum, na.rm = TRUE) / sapply(cpct, length), 1)
```

Cropland percentages:

- Clocolan (269) - 45%
- Ficksburg (274) - 39%
- Fouriesburg (275) - 29%
- Marquard (288) - 44%

10.1.3 Map selected districts, cropland percentage, and error

```r
brks1 <- seq(0, 100, 10)
cols1 <- rev(terrain.colors(length(brks1) - 1))
brks2 <- c(-100, -80, -60, -40, -20, -10, -5, -1)
brks2 <- c(brks2, rev(abs(brks2)))
cols2 <- colorRampPalette(c("red", "grey80", "blue4"))(length(brks2) - 1)
cols2[c(1, length(cols2))] <- c("darkred", "purple3")
cx <- 1.1
lcol <- "black"
mcap <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")

# layout.show()
pdf(fp(p_fig, "abm-selected-districts.pdf"), height = 7, width = 7)
par(mfrow = c(3, 2), mar = c(0, 0, 0, 0), oma = c(2, 0, 0, 0))
plot(md, col = "grey", border = "transparent")
plot(abm_md, col = "red", border = "grey30", add = TRUE)
par(mar = c(3, 0, 0, 0))
plot(abm_md)
plot(cb[[1]], axes = FALSE, box = FALSE, breaks = brks1, col = cols1,
     add = TRUE, legend = FALSE)
plot(abm_md, add = TRUE)
polygonsLabel(abm_md, labels = paste0(abm_md$NAME_2, " (", 1:4, ")"),
              method = "buffer", cex = 0.9)
for(i in 2:5) {
  plot(abm_md)
  plot(cb[[1]] - cb[[i]], axes = FALSE, box = FALSE, breaks = brks2,
       col = cols2, add = TRUE, legend = FALSE)
  plot(abm_md, add = TRUE, lwd = 2)

  mtext(mcap[i - 1], side = 1, line = -4, adj = 0.7)
}
nct <- length(brks1) - 1
flex_legend(ncuts = nct, legend.text = "% cropland", legend.vals = rep("", nct),
            longdims = c(0.6, 0.9), shortdims = c(0.72, 0.012), colvec = cols1,
            srt = c(270, 0), horiz = TRUE, textside = "bottom",
            legend.pos = c(4, 3), leg.adj = list(c(4.2, 0), c(-0.5, -0.5)),
            cex.val = 1.1, textcol = lcol, bordercol = lcol)
tc <- rect_coords(0.6, 0.9, nct, constEWorNS = 0.72, resEWorNS = 0.02)
xs <- c(tc[[1]][1, 1], sapply(tc, function(x) x[2, 1]))
ys <- sapply(tc, function(x) x[1, 2])
text(xs, ys, labels = brks1, srt = 270, adj = c(-0.15, 0.5), cex = 1.1)
nct <- length(brks2) - 1
flex_legend(ncuts = nct, legend.text = "% bias", legend.vals = rep("", nct),
            longdims = c(0.2, 0.8), shortdims = c(0.05, 0.015), colvec = cols2,
            srt = c(270, 0), horiz = TRUE, textside = "bottom",
            legend.pos = c(4, 6), leg.adj = list(c(4.2, 0), c(-0.5, -0.5)),
            cex.val = 1.1, textcol = lcol, bordercol = lcol)
tc <- rect_coords(0.2, 0.8, nct, constEWorNS = 0.047, resEWorNS = 0.012)
xs <- c(tc[[1]][1, 1], sapply(tc, function(x) x[2, 1]))
ys <- sapply(tc, function(x) x[1, 2])
text(xs, ys, labels = brks2, srt = 270, adj = c(-0.1, 0.5), cex = 1.1)
dev.off()
```

10.2 Select modeled yields for agent-based model

From earlier simulation of DSSAT yields (see rcropmod package), to provide information on yield variability in response to climate, results produced here for input into ABM

```
dssat <- fread(spathfunc("dssat-yields.csv", "ab"))
dssat_red <- copy(dssat)

dts <- c(2007288, 2007303, 2007318, 2007333, 2007348, 2007363, 2008013, 2008028)
dssat[PDAT %in% dts, ]

plot(1:31, 1:31, ylim = range(dssat$HWAH), pch = "", xlab = "YEAR",
     ylab = "HWAH")
points(dssat[PDAT %in% dts & FNAM != "ZASP0001", HWAH])
boxplot(dssat[PDAT %in% dts & FNAM == "ZASP0001", HWAH])
boxplot(dssat[PDAT %in% dts & FNAM == "ZASP0002", HWAH])
boxplot(dssat[PDAT %in% dts & FNAM == "ZASP0003", HWAH])
dssat[PDAT %in% dts, mean(HWAH), by = FNAM]
dssat[PDAT %in% dts, mean(HWAH), by = .(CU, FNAM)]

plot(1:8, 1:8, ylim = range(dssat$HWAH), pch = "", xlab = "YEAR",
     ylab = "HWAH")
for(i in unique(dssat$FNAM)) {
  dssat[PDAT %in% dts & CU == 1 & FNAM == i, lines(HWAH)]
}

# provincial ag census statistics from 2007
```

```
prodyld <- cbind.data.frame("md" = as.character(abm_md$NAME_2),
                            "pha" = c(6219, 6203, 6391, 9404),
                            "prod" = c(14781, 17434, 24366, 25578))
prodyld$yld <- prodyld$prod / prodyld$pha

outylds <- copy(dssat[PDAT %in% c(2007303, 2007348, 2008013) &
                   FNAM != "ZASP0001", ])
outylds[PDAT %in% dts, mean(HWAH), by = .(FNAM)]
outylds[PDAT %in% dts, mean(HWAH), by = CU]
outylds[PDAT %in% dts, mean(HWAH), by = .(CU, FNAM)]
write.csv(outylds, file = fp(p_data, "abm-yields.csv"))
```

10.3 Downscaling and agent allocation

Performed separately and described in the methods and supplemental.

10.4 Analysis of agent allocation errors

From output of ABM simulations

```
abias <- data.table(read_excel(spathfunc("stats.xlsx", "ab")))
setkey(abias, MD)
hhs <- data.table(read_excel(spathfunc("agent-bias2.xlsx", "ab"), sheet = 2))
setkey(hhs, MD)
abias <- abias[hhs]
setnames(abias, "geow", "glc")  # to account for mis-naming bug re. geowiki

# calculate district level bias as percent
pctbias <- abias[var == "ag_area",
                 lapply(list(sa30, globmu, modmu, glc), function(x) {
                   (gti - x) / gti * 100
                 })]

# HHs without land as percent total households
noland <- abias[var == "HHs_no_land",
                lapply(list(sa30, globmu, modmu, glc), function(x) {
                  x / HHs * 100
                })]

# unallocated farmland
unalloc <- round(abias[var == "ag_area_unalloc", snms, with = FALSE] /
  abias[var == "ag_area", snms, with = FALSE] * 100, 2)

# land deficit farmland
land_def <- round(abias[var == "land_def", snms, with = FALSE] /
  abias[var == "ag_area", snms, with = FALSE] * 100, 2)

# households couldn't find enough land
not_enough <- abias[var == "HHs_cant_find", snms, with = FALSE] /
                      (abias[var == "HHs_no_land",
                             lapply(.SD, function(x) HH_gen - x),
                             .SDcol = c(snms, "HH_gen")][, snms, with = FALSE])
```

```r
not_enough <- round(not_enough, 2)

# production deficits
hhprod <- abias[var == "prod", gti / HHs]
mdprodlc <- abias[var == "prod",
                  lapply(.(sa30, globmu, modmu, glc), function(x) x / HHs)]
prdef <- abias[var == "prod", lapply(.(sa30, globmu, modmu, glc), function(x) {
  (gti - x) / gti * 100
})]  # district total deficit
hhprodlc <- abias[var == "prod",
                  lapply(.(sa30, globmu, modmu, glc), function(x) x / HHs)]
hhprod_def <- (hhprod - hhprodlc) / hhprod * 100  # household deficit

cexs <- seq(1, 2, 1 / 3)
pdf(fp(p_fig, "fig6.pdf"), height = 2.5, width = 5.25)
cx <- c(0.9, 1, 0.6)
par(mfrow = c(1, 3), mar = c(4, 2.5, 0.5, 0), mgp = c(1.1, 0.1, 0),
    tcl = 0.2, oma = c(3.5, 0, 0, 0.1))
scols <- c("red", "orange3", "green4", "blue")
cols <- c(rep("red", 4), rep("orange3", 4), rep("green4", 4), rep("blue", 4))
plot(unlist(pctbias), unlist(unalloc), col = cols, cex = cexs, pch = 20,
     ylab = "% land unallocated", xlab = "% error in cropland area",
     cex.lab = cx[2], cex.axis = cx[1])
mtext("(a)", side = 3, adj = -0.27, line = -1)
plot(unlist(pctbias), unlist(land_def), col = cols, cex = cexs, pch = 20,
     ylab = "% land deficit", xlab = "% error in cropland area",
     cex.lab = cx[2], cex.axis = cx[1])
mtext("(b)", side = 3, adj = -0.23, line = -1)
plot(unlist(pctbias), unlist(hhprod_def), col = cols, cex = cexs, pch = 20,
     ylab = "% food deficit", xlab = "% error in cropland area",
     cex.lab = cx[2], cex.axis = cx[1])
mtext("(c)", side = 3, adj = -0.23, line = -1)
par(xpd = NA)

yst <- seq(0.05, 0.23, 0.18 / 4)
xy <- cbind(grconvertX(rep(0.5, 5), from = "ndc"),
            grconvertY(yst, from = "ndc"))
for(i in 2:length(yst)) {
  if(i == 5) {
    leg <- c("District", 1:4)
  } else {
    leg <- rep("", 4)
  }
  legend(x = xy[i, 1], y = xy[i, 2], legend = rep("",4), #legend = leg,
         pch = 20, cex = cx[3],
         horiz = TRUE, adj = c(4, -5.75),
         col = scols[i - 1], pt.cex = cexs, bty = "n")#cex = cexs,)
}
yst <- seq(0.06, 0.24, 0.18 / 4)
xy <- cbind(grconvertX(rep(0.505, 5), from = "ndc"),
            grconvertY(yst, from = "ndc"))
text(x = xy[, 1], y = xy[, 2], labels = c(mcap, "District"),
     #srt = 90, #pos = 2,
     #adj = c(-1, 1),
     pos = 2, col = c(scols, "black"), cex = cx[1])
xst <- seq(0.53, 0.60, 0.07 / 3)
xy <- cbind(grconvertX(xst, from = "ndc"),
            grconvertY(rep(yst[length(yst)], 4), from = "ndc"))
text(x = xy[, 1], y = xy[, 2], labels = 1:4, pos = 2, cex = cx[1])
dev.off()
```

11 Cropland error impact on locational accuracy

Evaluated in terms of how landcover map error impacts the ability to locate areas having a particular quantity of interest.

11.1 Data

Derived carbon (section 6) and yield estimates (section 8).

```
library(croplandbias)
library(xtable)
library(RColorBrewer)
library(spatstat)

p_root <- fp(proj_root("croplandbias"), "croplandbias")
p_fig <- fp(p_root, "inst/paper/figures/")
p_data <- fp(p_root, "inst/extdata/")

# SA provinces
data(sashp)
data(prov)
prov <- prov[prov$id_1 != 9, ]  # remove Princeton Edward Islands
prov[prov$id_1 == 10, "id_1"] <- 9 # reset WC to id 10

# yield and carbon data
load(spathfunc("yieldprod-1km.rda", "yl"))
load(spathfunc("carbon-tier1.rda", "cb"))
load(spathfunc("cropland-1km.rda"))
```

11.2 Differences in location

If you are interested in selecting areas of highest yield/production/carbon potential, how far off are you relative to "truth". Use top quintile/decline to assess this, and then calculate mean nearest neighbor distance between two pixels of each map.

11.2.1 Identify high value areas

```
# top decile of...
# ...cropland
crclass <- lapply(cropl, function(x) {  # x <- cropl$sa30
  r <- x * (x > 0)
```

```r
  r[r == 0] <- NA
  rq <- quantile(r, c(0, 0.9, 1))
  types <- cut(r, breaks = rq)
  types > 1
})

# ...yield
yclass <- lapply(cyld_agg$f1, function(x) {  # x <- cyld_agg$f1[[1]]
  r <- x * (x > 0)
  r[r == 0] <- NA
  rq <- quantile(r, c(0, 0.9, 1))
  types <- cut(r, breaks = rq)
  types > 1
})

# ...production
pclass <- lapply(cpagg1$f1, function(x) {  # x <- cyld_agg$f1[[1]]
  r <- x * (x > 0)
  r[r == 0] <- NA
  rq <- quantile(r, c(0, 0.9, 1))
  types <- cut(r, breaks = rq)
  types > 1
})

# ...carbon
cdat <- c(gti_c$f1, lc_c$f1)
cdatf <- lapply(cdat, function(x) {  # mask out non-cropland areas and take mu C
  msk <- x$forest < 99.6
  cmu <- calc(x, mean)
  raster::mask(cmu, msk, maskvalue = 0)
})
cclass <- lapply(cdatf, function(x) {  # x <- cyld_agg$f1[[1]]
  r <- x * (x > 0)
  r[r == 0] <- NA
  rq <- quantile(r, c(0, 0.9, 1))
  types <- cut(r, breaks = rq)
  types > 1
})
names(cclass) <- names(pclass)

# carbon efficiencies (C / yield)
effclass <- lapply(1:5, function(x) { # x <- 1
  cyld <- cyld_agg$f1[[x]]
  cyld[cyld == 0] <- NA
  cdatfc <- crop(cdatf[[x]], cyld)
  eff <- cdatfc / cyld  # carbon efficiency
  # eff[is.infinite(eff)] <- NA
  rq <- quantile(eff, c(0, 0.9, 1))
  types <- cut(eff, breaks = rq)
  types > 1
})
names(effclass) <- names(cclass)
```

11.2.2 Calculate nearest-neighbor distances

Between GTI high value areas and each of the four LC-derived datasets.

```r
# calculate distance between GTI and each other dataset's classes.
# pms2a <- expand.grid(rep(list(0:1), 5))[-1, ]
# pms2b <- pms2a[which(rowSums(pms2a) == 2 & pms2a$Var1 == 1), ]
# pms2b$cl <- c("YC1", "YC2", "YC3", "YC4")
# rownames(pms2b) <- 1:4

# Define neighborhood window
sa_owin <- owin(xrange = bbox(sashp)[1, ], yrange = bbox(sashp)[2, ],
                unitname = "meter")
# which1 <- function(x) x == 1  # selector function for r to xy

mudists <- lapply(list(crclass, yclass, pclass, cclass, effclass), function(x) {
  # x <- cclass
  mudist <- sapply(2:5, function(y) {  # y <- 2
    print(paste0("...", y))
    a <- rasterToPoints(x$gti, fun = function(x) x == 1)
    b <- rasterToPoints(x[[y]], fun = function(x) x == 1)
    pps <- lapply(list(a, b), function(z) {
      ppp(x = z[, 1], y = z[, 2], window = sa_owin)
    })
    names(pps) <- c("gti", names(x)[y])
    reord <- sort(sapply(pps, function(i) i$n), decreasing = TRUE)
    ppnn <- nncross(pps[[names(reord)[1]]], pps[[names(reord)[2]]])
    mean(ppnn$dist) / 1000
  })
  names(mudist) <- names(x)[2:5]
  mudist
})
names(mudists) <- c("cropland", "yield", "prod", "carb", "eff")

# average km off
mudvec <- unlist(mudists)
median(mudvec[!like(names(mudvec), "sa30")])
# mean(mudvec)

# plot(stack(yclass))
# plot(is.na(yclass[[1]]) - is.na(yclass[[2]]))
# plot(is.na(cyld_agg$f1$gti) - is.na(cyld_agg$f1$glc))

# output plot
cols <- c("red", "orange3", "green4", "blue")
pcol <- makeTransparent(cols, alpha = c(120))
pcol2 <- makeTransparent(cols, alpha = c(30))

# plot(1:4, pch = 20, col = pcol)
bs <- do.call(rbind, mudists[c(1, 2, 4)])
lcnms <- c("SA-LC", "GlobCover", "MODIS", "GLC-Share")

pdf(full_path(p_fig, "fig7.pdf"), height = 3.5, width = 4,
    bg = "transparent")
```

```r
par(mar = c(2, 3.5, 1, 0), bg = "transparent")
bp <- barplot(bs, beside = TRUE, col = ggplot2:::interleave(cols, pcol, pcol2),
              las = 2, border = "transparent", xaxt = "n", ylab = "",
              ylim = c(0, 18), yaxt = "n")
axis(1, at = colMeans(bp), labels = lcnms, lty = 0, cex.axis = 0.9,
     mgp = c(2, 0.25, 0))
axis(2, at = seq(0, 18, 2), labels = seq(0, 18, 2), las = 2, mgp = c(2, 0.2, 0),
     tcl = 0.4)
mtext("km", side = 2, line = 2)
legend(x = 12.5, y = 18, legend = rep("", 4), pch = 15, col = cols, adj = 0,
       pt.cex = 1.5, bty = "n", cex = 0.8, x.intersp = 0, horiz = TRUE)
legend(x = 12.5, y = 17.25, legend = rep("", 4), pch = 15, adj = 0, col = pcol,
       pt.cex = 1.5, bty = "n", cex = 0.8, horiz = TRUE, x.intersp = 0)
legend(x = 12.5, y = 16.5, legend = rep("", 4), pch = 15, adj = 0, col = pcol2,
       pt.cex = 1.5, bty = "n", cex = 0.8, horiz = TRUE, x.intersp = 0)
text(x = 13, y = 17, labels = "Cropland", pos = 2, cex= 0.8)
text(x = 13, y = 16.25, labels = "Yield", pos = 2, cex = 0.8)
text(x = 13, y = 15.5, labels = "Carbon", pos = 2, cex= 0.8)
dev.off()
```

12 Miscellaneous additional analyses

A few additional calculations and figures undertaken for the paper, including supplemental figure (requested by final review) of example illustration of different land-cover datasets analyzed

```r
library(croplandbias)
library(rgeos)
library(dismo)
library(sf)
library(viridis)

rm(list = ls())
p_root <- fp(proj_root("croplandbias"), "croplandbias")
p_edat <- fp(p_root, "external/ext_data")
p_fig <- fp(p_root, "inst/paper/figures/")
```

12.1 South Africa's share of sub-Saharan Africa's area

```r
eco <- readOGR(dsn = fp(p_edat, "africa_ecofloristic_zones.sqlite"),
               layer = "africa_ecofloristic_zones", verbose = FALSE)
crs(eco) <- crs(projection(raster(nrow = 1, ncol = 1)))
data(africa)
ecoalb <- spTransform(eco, crs(africa))

# cut down countries
nms <- africa$cntry_name[africa$region == "Northern Africa"]
nms <- nms[nms != "Sudan"]  # keep Sudan
ssa <- africa[!africa$cntry_name %in% nms, ]
ssa <- ssa[-grep("Island|Sao|Verde|Mauritius|Helena|Comoros|Portugal|Seychell",
                 as.character(ssa@data$cntry_name)), ]
```

```r
ssa <- gBuffer(ssa, width = 0)
# plot(ssa)

desert <- ecoalb[grep("desert", ecoalb$gez_term), ]
# plot(ecoalb[names(which.max(gArea(desert, byid = TRUE) / 10000 / 100)), ])
sahara <- as.numeric(names(which.max(gArea(desert, byid = TRUE))))

nosahara <- ecoalb[-sahara, ]
# plot(nosahara, col = "red")

ovs <- which(!is.na(over(nosahara, ssa)))
ssaeco <- nosahara[ovs, ]
ssaeco <- ssaeco[-1, ]  # drop a couple Saharan mountains
ssaeco <- ssaeco[-1, ]
# plot(ssaeco[ssaeco@data$gez_term == "Tropical mountain system", ][1, ])
# plot(ssa)
# plot(ssaeco, col = "red", add = TRUE)

# calculate area of remainder
afarea <- gArea(ssaeco) / 10000
# plot(af[af$cntry_name == "South Africa", ])
saarea <- gArea(africa[africa$cntry_name == "South Africa", ]) / 10000
saarea / afarea
```

```
## [1] 0.05771493
```

12.2 Calculation of error propagation

```r
load(spathfunc("cropland-werr-2011.rda"))
load(spathfunc("yield-prod-accbias.rda", "yl"))
load(spathfunc("et-err-bias.rda", "et"))
load(spathfunc("carbon-werr.rda", "cb"))

cland <- bacc2011
yprod <- as.data.table(out_tab)[Region == "Density"]
carb <- rbind(cb_statsw_mu, cb_statsw_mua)

# Reshape error stats
# yield/production
yp1 <- melt(yprod, id.vars = c("Map", "Metric", "Variable"),
            measure.vars = names(yprod)[5:10])
yp2 <- dcast(yp1, formula = Metric + Variable + variable ~ Map)
setcolorder(yp2, neworder = c(1:3, ncol(yp2), 5:6, 4))
setnames(yp2, names(yp2), c("stnm", "var", "ol", names(cland)[-c(1:2)]))

snms <- names(cland)[-c(1:2)]
lev <- cland[stnm == "mu", ol]

# Production and yield propagation
yp_prop <- rbindlist(lapply(c("Yield", "Production"), function(x) {
  o <- data.frame(t(sapply(snms, function(y) {  # y <- "sa30"
    emag <- yp2[var == x & stnm == "MAE", get(y)] /
```

```
      cland[stnm == "mua", get(y)]
    c(emag, mean(emag))
  })))
  o <- cbind(snms, round(o, 2))
  colnames(o) <- c("map", lev, "mu")
  data.table(cbind.data.frame("var" = x, o))
}))

# Carbon differences
c_prop <- rbindlist(lapply(snms, function(x) {  # x <- "sa30"
  dat <- carb[stnm == "MAE" & map == x, names(carb)[-c(1:3)], with = FALSE]
  o <- dat[, lapply(.SD, function(y) y / cland[stnm == "mua", get(x)]),
           .SDcols = names(dat)]
  o <- round(rbind(o, data.table(t(colMeans(o)))), 2)
  o <- cbind("ol" = c(lev, "mu"), o)
  cbind("map" = x, o)
  # dat[, second] / cland[stnm == "mua", get(x)]
}))

# et propagation
et_prop <- round(et_err[Variable == "Annual Mean", MAE] /
  cland[ol == "f25" & stnm == "mua", snms, with = FALSE], 2)

yp_prop
```

```
##             var    map   f1   f5  f10  f25  f50 f100   mu
## 1:        Yield   sa30 0.11 0.03 0.04 0.04 0.09 0.14 0.08
## 2:        Yield globmu 0.20 0.04 0.03 0.06 0.09 0.10 0.09
## 3:        Yield  modmu 0.54 0.39 0.30 0.26 0.27 0.20 0.33
## 4:        Yield    glc 0.34 0.25 0.18 0.24 0.22 0.29 0.25
## 5: Production   sa30 1.71 1.57 1.42 1.20 0.76 0.37 1.17
## 6: Production globmu 1.98 3.11 3.48 3.57 3.11 2.59 2.97
## 7: Production  modmu 1.85 2.63 2.84 2.67 1.81 1.24 2.17
## 8: Production    glc 2.00 3.32 3.73 3.88 3.48 1.91 3.05
```

```
c_prop
```

```
##         map   ol forest second shrub grass sparse  All
##   1:   sa30   f1   3.13   2.47  2.40  0.04   0.61 1.73
##   2:   sa30   f5   2.30   1.96  1.92  0.04   0.64 1.37
##   3:   sa30  f10   2.11   1.83  1.79  0.04   0.65 1.28
##   4:   sa30  f25   1.83   1.63  1.60  0.04   0.67 1.15
##   5:   sa30  f50   1.57   1.43  1.41  0.04   0.69 1.03
##   6:   sa30 f100   1.45   1.33  1.31  0.04   0.71 0.97
##   7:   sa30   mu   2.07   1.77  1.74  0.04   0.66 1.26
##   8: globmu   f1   5.77   3.90  3.73  0.04   0.55 2.80
##   9: globmu   f5   3.06   2.47  2.41  0.04   0.60 1.72
##  10: globmu  f10   2.61   2.19  2.14  0.04   0.61 1.52
##  11: globmu  f25   2.12   1.86  1.82  0.04   0.64 1.29
##  12: globmu  f50   1.78   1.60  1.57  0.04   0.66 1.13
##  13: globmu f100   1.54   1.41  1.39  0.04   0.69 1.01
##  14: globmu   mu   2.81   2.24  2.18  0.04   0.62 1.58
##  15:  modmu   f1   4.68   3.30  3.17  0.04   0.57 2.35
##  16:  modmu   f5   2.38   2.02  1.97  0.04   0.63 1.41
##  17:  modmu  f10   2.09   1.82  1.78  0.04   0.65 1.27
## 18:  modmu  f25   1.82   1.62  1.60  0.04   0.66 1.15
## 19:  modmu  f50   1.62   1.46  1.44  0.04   0.69 1.05
## 20:  modmu f100   1.46   1.34  1.32  0.04   0.71 0.97
## 21:  modmu   mu   2.34   1.93  1.88  0.04   0.65 1.37
## 22:    glc   f1   3.82   2.81  2.72  0.04   0.60 2.00
## 23:    glc   f5   2.28   1.94  1.90  0.04   0.64 1.36
## 24:    glc  f10   2.08   1.80  1.77  0.04   0.66 1.27
## 25:    glc  f25   1.85   1.64  1.61  0.04   0.67 1.16
## 26:    glc  f50   1.59   1.44  1.42  0.04   0.69 1.04
## 27:    glc f100   1.42   1.30  1.29  0.04   0.71 0.95
## 28:    glc   mu   2.17   1.82  1.78  0.04   0.66 1.30
##        map   ol forest second shrub grass sparse  All
```

```
et_prop
```

```
##    sa30 globmu modmu  glc
## 1: 0.09   0.04  0.04 0.07
```

12.3 Land cover data illustrated

12.3.1 Read in base land cover in Albers format

For this part, steps taken to get to Albers-transformed versions of base land cover data subset to South Africa's extent will have to be repeated. Due to size, these were not saved in the repo. See chapter 1 and chapter 3 to recreate these initial steps.

```
sa_r <- raster(spathfunc("sagrid.tif"))
glc <- raster(fp(p_edat, "landcover/glc_share/glcSA_alb_rect.tif"))
glob <- raster(fp(p_edat, "landcover/globcover_2009/globSA_alb.tif"))
modis <- raster(fp(p_edat, "landcover/MCD12Q1/modis_type_1_alb.tif"))
salc <- raster(fp(p_edat, "landcover/sa_lc_2009/sa_lc_2009.tif"))

# Also read in raw shapefile of South Africa field boundaries, for illustration
gti <- st_read(dsn = fp(p_edat, "landcover/gti/GTI_SA_2011.shp"),
               layer = "GTI_SA_2011")

# small image of South Africa, in GCS
ext <- extent(c(26.95, 27.35, -28.72, -28.43))  # arbitrary small cropland area
base <- gmap(ext, type = "satellite", lonlat = TRUE, rgb = TRUE)
base <- trim(base)
baseext <- as(extent(base), "SpatialPolygons")
gcs <- data.table(make_EPSG())[like(code, "4326"), prj4]
crs(baseext) <- CRS(gcs)
baseext_alb1 <- spTransform(baseext, crs(sashp))  # albers used for paper
baseext_alb2 <- spTransform(baseext, crs(salc))  # albers used for native SA-LC
```

12.3.2 Crop them down to save

```
gtic <- st_intersection(gti, st_as_sf(baseext))  # crop GTI layers
salcc <- crop(salc, baseext_alb2)  # crop with native albers extent
globc <- crop(glob, baseext_alb1)  # crop with albers used in paper
modisc <- crop(modis, baseext_alb1)  # ""
glcc <- crop(glc, baseext_alb1)  # ""
```

```r
# Set minor (<10 pixels) cover types to NA
# hist(salcc, maxpixels = ncell(salcc), plot = FALSE, breaks = seq(0.5, 5.5, 1))
# only 6 pixels in degraded in degraded class, so plays havoc with levelplot and
# ggplot--set to NA
salcc[salcc == 3] <- NA

# table(values(modisc))  # class 9 and 13 only have 2 and 6 each. Set both to NA
modisc[modisc %in% c(9, 13)] <- NA

# table(values(globc))  # class 14 (rainfed cropland), 60, and 130 have 1, 8, 2
globc[globc %in% c(14, 60, 130)] <- NA

# add categories to categorical data
# cols <- rev(terrain.colors(8))
# cols <- c("khaki")

# display.brewer.all()
crcols <- brewer.pal(11, "PuOr")[3:5]
blth20 <- brewer.pal(11, "Paired")[6]
natveg <- brewer.pal(9, "YlGn")[c(2, 3, 4)]
# plot(1:5, pch = 20, col = natveg, cex = 3)

# SA-LC
salcc <- ratify(salcc)
rat <- levels(salcc)[[1]]
rat[["lc"]] <- c("Natural", "Cultivation", "Built", "Water")
rat[["cols"]] <- c(natveg[3], crcols[1], blth20, "blue")
levels(salcc) <- rat

# GlobCover
globc <- ratify(globc)
rat <- levels(globc)[[1]]
rat[["lc"]] <- c("Mosaic veg/crop", "Mosaic for-shrub/grass",
                 "Closed/open herb", "Sparse veg", "Water")
rat[["cols"]] <- c(crcols[2], rev(natveg), "blue")
levels(globc) <- rat

# MODIS
modisc <- ratify(modisc)
rat <- levels(modisc)[[1]]
rat[["lc"]] <- c("Open shrub", "Grasslands",
                 "Croplands", "Cropland/nat. veg")
rat[["cols"]] <- c(natveg[2:1], crcols[1], crcols[2])
levels(modisc) <- rat
```

12.3.3 And plot

```r
sasf <- st_as_sf(spTransform(sashp, CRS(gcs)))  # sasp to sf (gcs)

l <- -1
cx <- 0.8
png(fp(p_fig, "lc-demo-fig.png"), width = 6, height = 2.5, units = "in",
    res = 300)
par(mfrow = c(2, 4), mar = c(0, 0, 0, 1.5), oma = c(0, 0, 0, 0))

# sa shp
plot(st_geometry(sasf), col = "grey")
plot(st_as_sf(baseext), col = "red", add = TRUE)

# Base image
plot(baseext, lty = 0)
plotRGB(base, add = TRUE)

# Base image with GTI polys
plot(baseext, lty = 0)
plotRGB(base, add = TRUE)
mtext("Reference", side = 3, line = 1, cex = cx)
plot(st_geometry(gtic), border = "transparent", col = crcols[1], add = TRUE)

# GLC-Share
aargs <- list(mgp = c(1, 0.25, 0), at = seq(0, 100, 10),
              tcl = -0.2, cex.axis = 0.6)
plot(baseext_alb1, lty = 0)
plot(glcc, add = TRUE, axes = FALSE, legend = FALSE, box = FALSE,
     interpolate = FALSE, col = inferno(10), breaks = seq(0, 100, 10))
mtext("GLC-Share", side = 3, line = 1, cex = cx)
plot(glcc, legend.only = TRUE, title = "%", legend.width = 1, col = inferno(10),
     legend.shrink = 0.5, axis.args = aargs,
     legend.args = list("%", cex = 0.6))

# SA-LC, GlobCover, MODIS
plot(baseext_alb2, lty = 0)
plot(salcc, add = TRUE, axes = FALSE, legend = FALSE, box = FALSE,
     col = levels(salcc)[[1]]$col)  # SA-LC
mtext("SA-LC", side = 3, line = 1, cex = cx)
plot(baseext_alb1, lty = 0)
plot(globc, add = TRUE, axes = FALSE, legend = FALSE, box = FALSE,
     col = levels(globc)[[1]]$col)  # Glob-Cover
mtext("GlobCover", side = 3, line = 1, cex = cx)
plot(baseext_alb1, lty = 0)
plot(modisc, add = TRUE, axes = FALSE, legend = FALSE, box = FALSE,
     col = levels(modisc)[[1]]$col)  # MODIS
mtext("MODIS", side = 3, line = 1, cex = cx)

# Legend
legtxt <- c(" NAT / FSG /  -  ",
            "  -  / COH / OSH ",
            "  -  / SVG / GRL ",
            " CUL /  -  / CLD",
            "  -  / MVC / CNV ",
            " BLT /  -  /  - ",
            " H2O / H2O /  -  ")
plot(baseext, lty = 0)
par(family = "mono")
legend("left", bty = "n", legend = legtxt, title = "SA-LC/GlobCover/MODIS",
       fill = c(rev(natveg), crcols[1:2], blth20, "blue"),
       inset = -0.005, y.intersp = 0.7, x.intersp = 0.5, cex = 0.8,
       title.adj = 3.5)

dev.off()
```