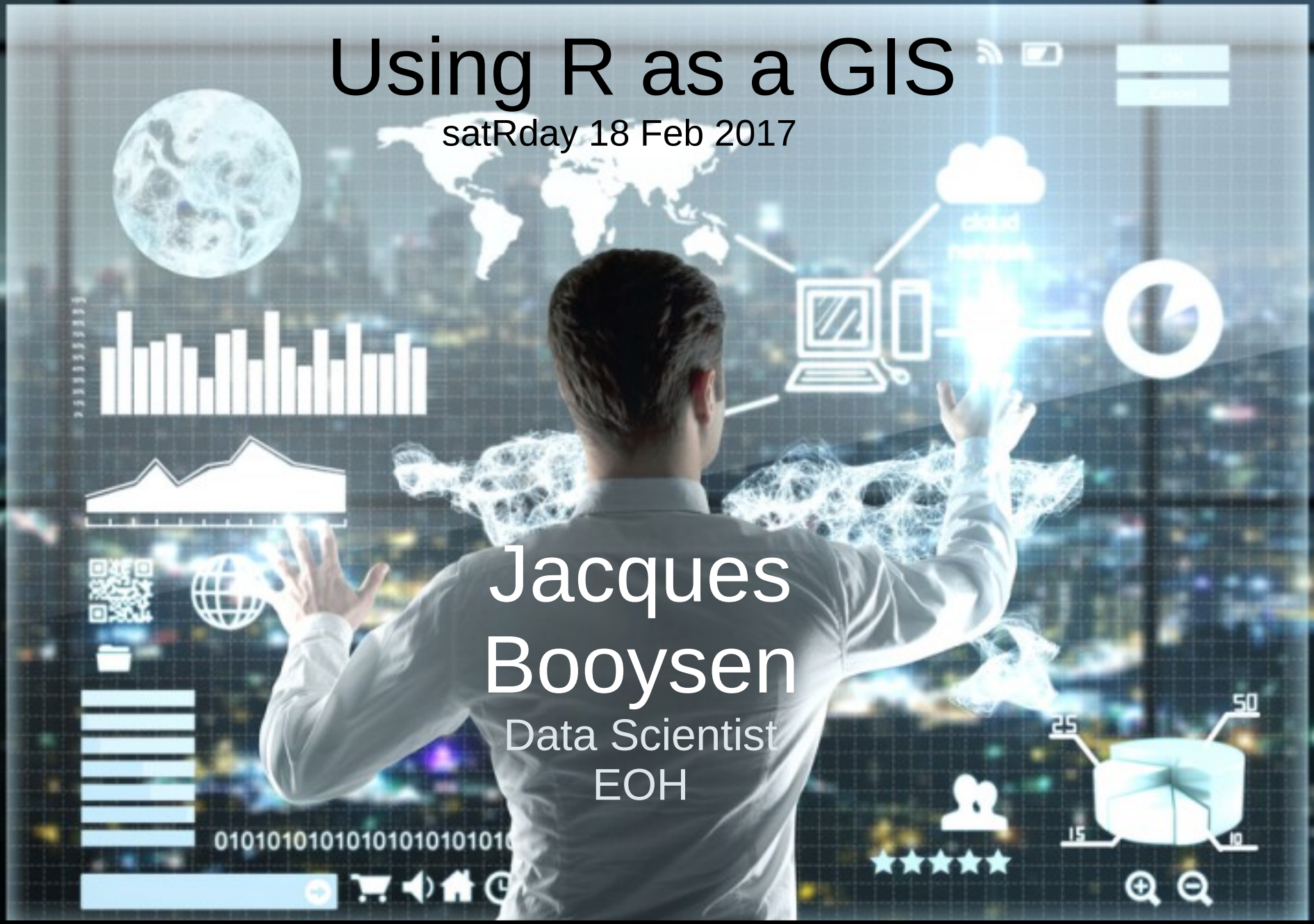
Using R as a GIS

satRday 18 Feb 2017



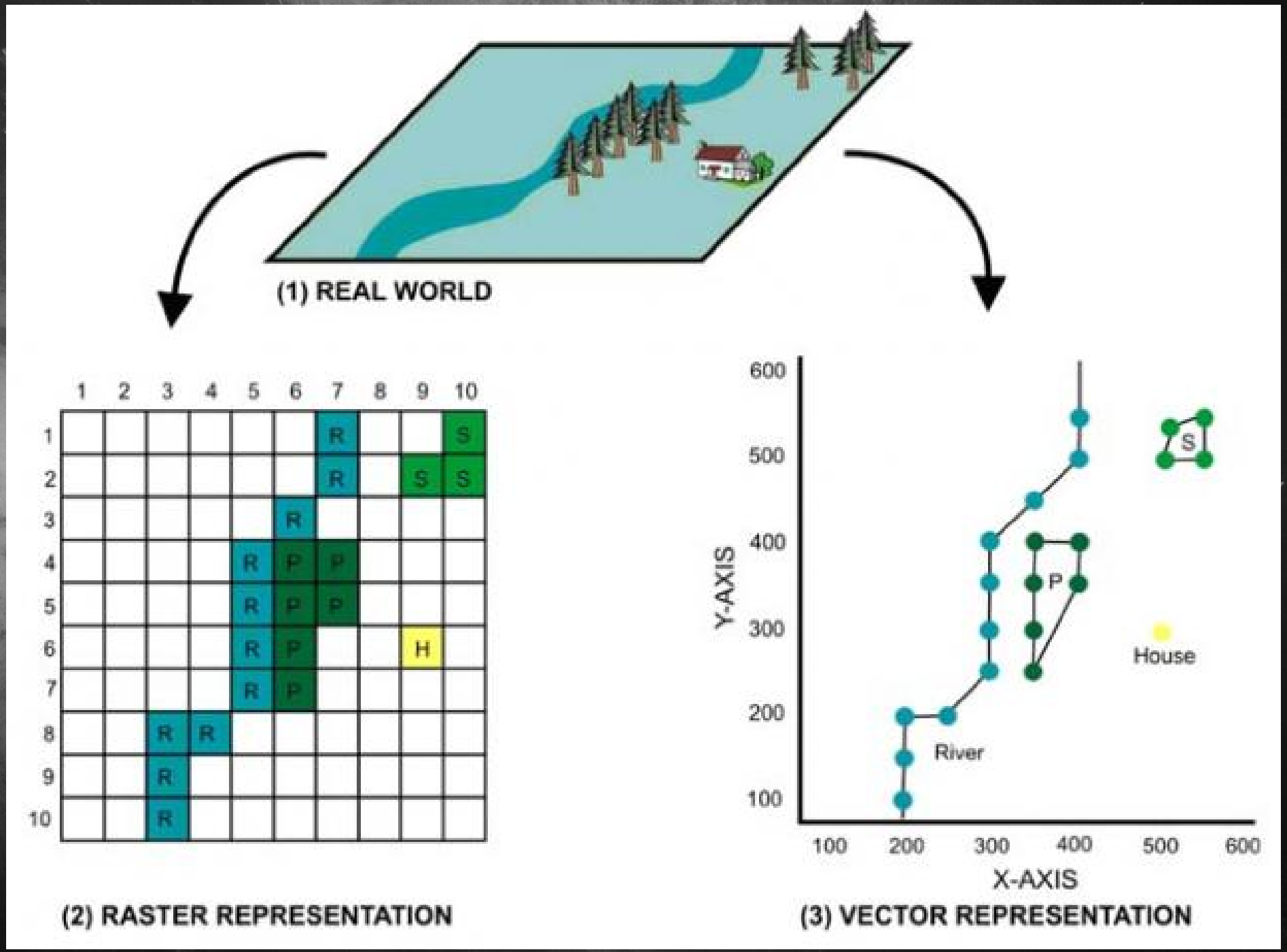
Jacques Booysen

Data Scientist

EOH

Practical applications using R for spatial data visualization,creation and manipulation

GIS data overview



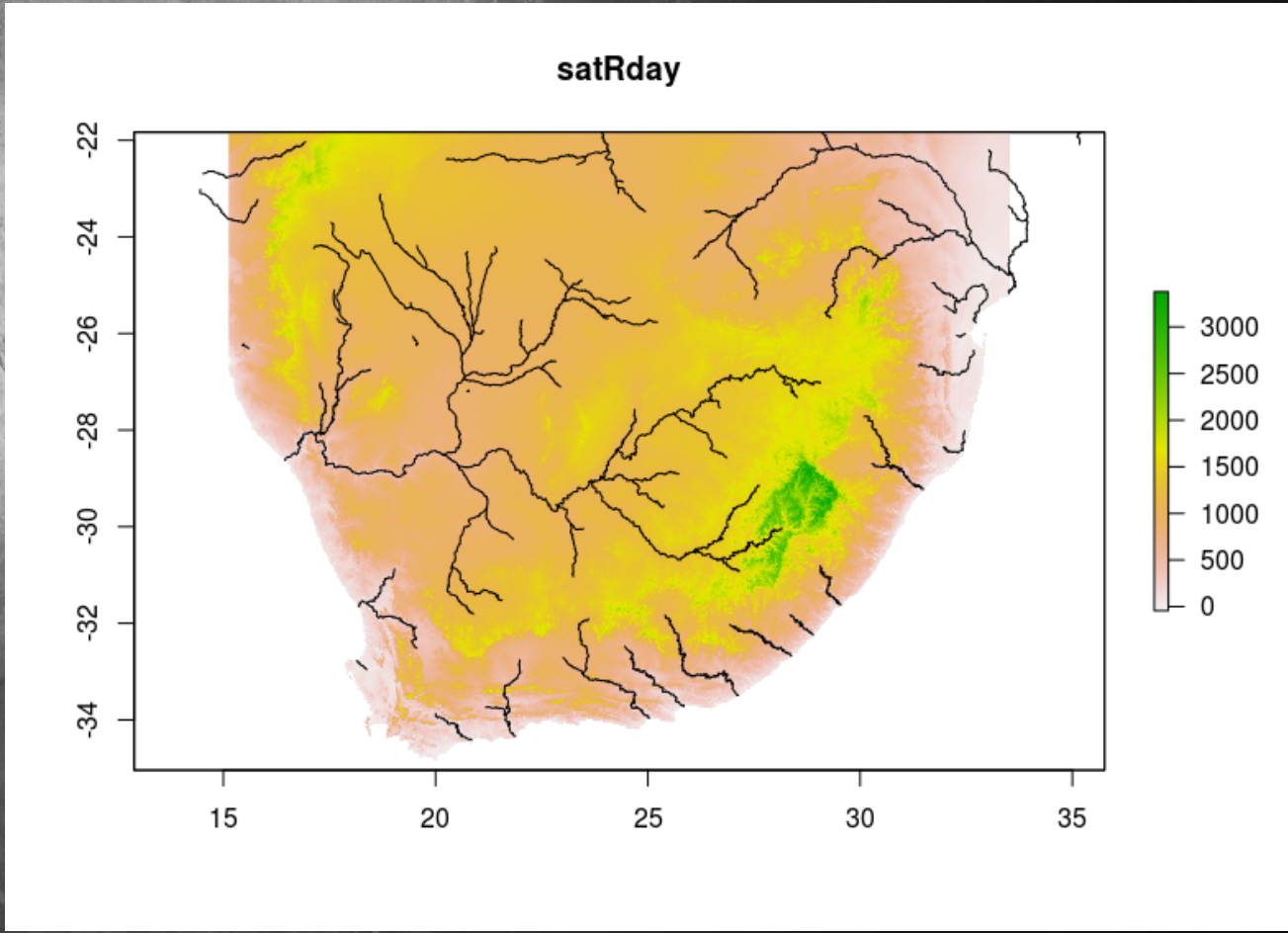
R GIS Package Roadmap (Intro)

- sp, maptools
  - Tools for loading and using spatial data including shapefiles (vectors).
- maps
  - Easy to use map polygons for plots.
- ggmap
  - Download street maps straight from Google maps and use them as a background in your ggplots.
- raster
  - Reading, writing, manipulating, analyzing and modeling of gridded spatial data. The package implements basic and high-level functions. Processing of **very large** files is supported.
- rgdal:
  - This package provides methods for working with importing and exporting different **raster** and **vector** geospatial data formats; Coordinate Reference Systems; projections, etc.
  -

Visualise GIS data in R

Example: Visualise Raster/Vector GIS Data in R

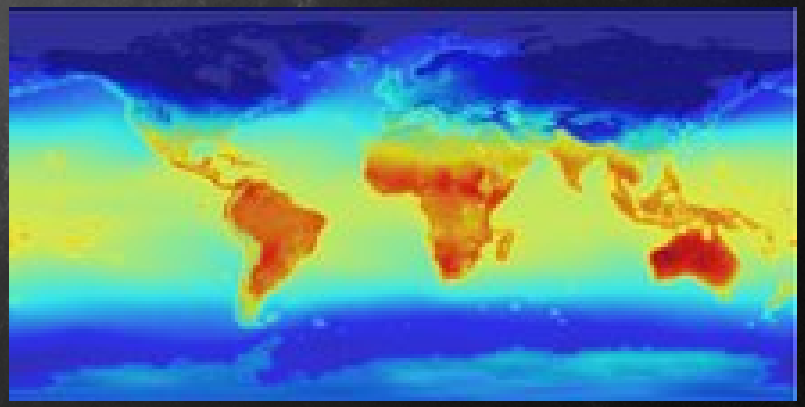
```
library(raster)
library(rgdal)

# Read and plot Raster grid
maps.
#   GeoTiff: Digital elevation
model in this case...
sa = readGDAL("southafrica.tif")
rsa = raster(sa,layer=1)

# Read and plot Vector
#    shapefile. Rivers
rivers = readOGR(
“rivers_subset.shp",
    layer="rivers_subset")

plot(rsa,main="satRday")
plot(rivers,add=T)
```



Manipulate GIS data in R

Example: Manipulate NASA climate change data on Amazon Web Services S3 volumes

- Data & Size: NEX Global Daily Downscaled Climate Projections **12 TB**
- Each of the climate projections includes **daily** maximum temperature, minimum temperature, and precipitation for the periods from 1950 through 2100. The spatial resolution of the dataset is 0.25 degrees (~25 km x 25 km).



Manipulate GIS data in R

- Used a climate change masterdata file to one by one copy chucks of +-1GB files very fast over AWS S3 into a temp directory and process them on an c4.8xlarge EC2 instance, stepping through all 12 TB data but only stripping out the GIS data relevant to my experiment using R.
- These files are multi band raster files with each band representing future temperature/rainfall values for each day of the year for all grid positions 100 years into the future across multiple climate scenarios or climate change pathways.
- Made use of the raster package to read these multi-band GIS raster files into memory, manipulate them by Cropping them per country and Rolling them up to weekly values to make them smaller and easy to process.

Manipulate GIS data in R

Country Vector masking layer

- africa = readOGR(dsn = "AfricanCountires.shp",layer = "AfricanCountires")

Use brick instead of raster to load multiband raster files into memory.

theraster = **brick**(“climatemodel.nc”,varname=”pr")

 # Rainfall

theraster <- **crop**(theraster,

Crop First because it is FASTER than mask

extent(africa[africa@data$COUNTRY==”South Africa”,]))

theraster <- **mask**(thec, africa[africa@data$COUNTRY==”South Africa”,])

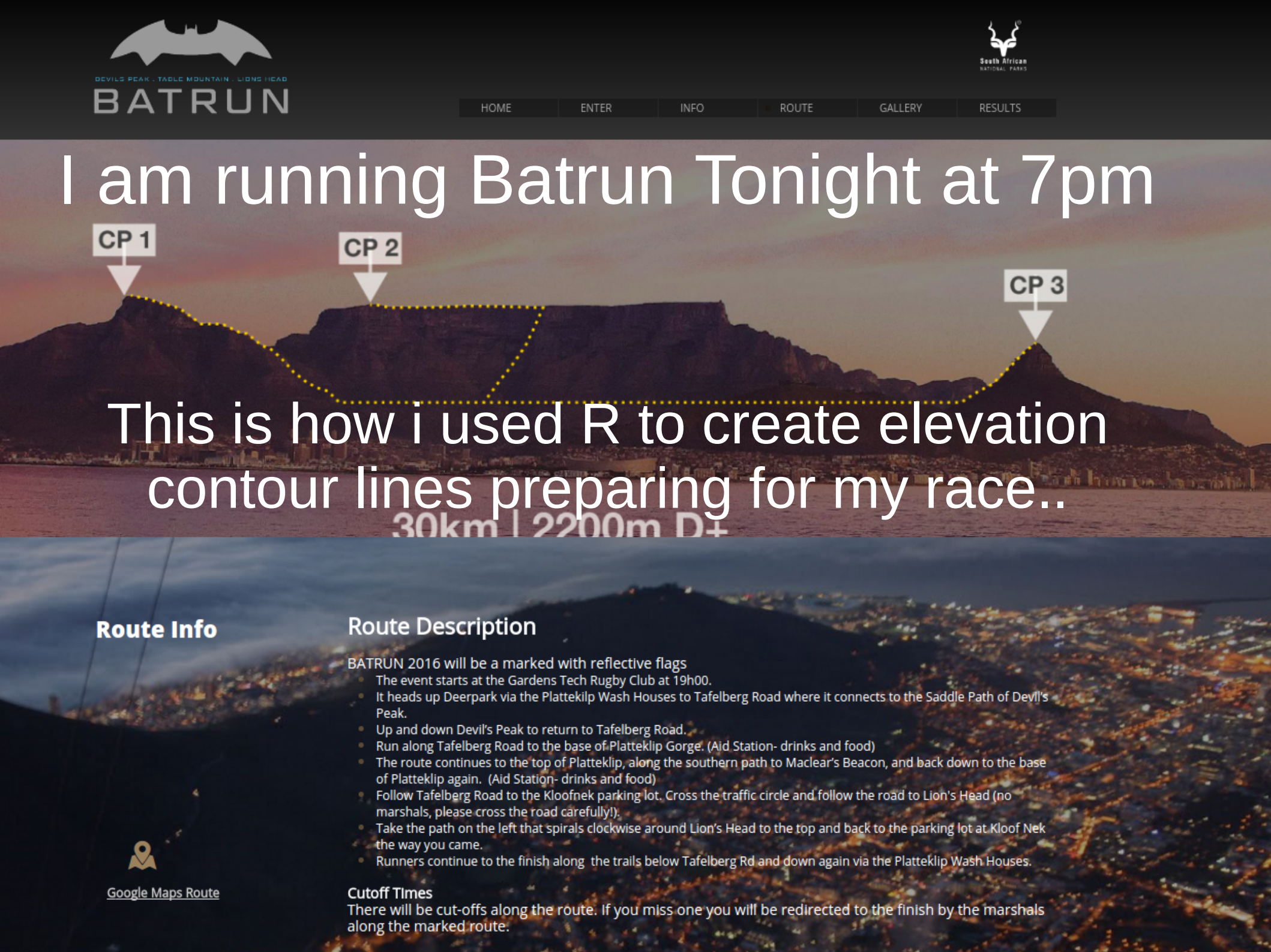
Manipulate GIS data in R

```
thedates = names(theraster)
thedates = as.Date(gsub("\\.","-",substr(thedates,2,11)))
theweeks =  unique(week(thedates)) # package: lubridate
######## RollUp Daily GIS data to weekly data #######
thedat = do.call(rbind,lapply(theweeks,function(w) {
      thew =  theraster[[ c(1:length(thedates))
                          [week(thedates)==w] ]]
              cbind(week=w,mean(values(thew),na.rm=T) )
         }))
```

Creating GIS data in R

Example: Create topographic contour lines as GIS vectors using elevation data from google API.

- Input Data: **Google Elevation API**
  - Provides elevation data for all locations on the surface of the earth, including depth locations on the ocean floor (negative values)
  - 2500 free requests per day :)
  - Sampled Path Requests (example: 'path'='36.578581,-118.291994|36.23998,-116.83171')
    - **path** (required) defines a path on the earth for which to return elevation data. This parameter defines a set of two or more ordered {latitude,longitude} pairs defining a path along the surface of the earth.
    - **samples** (required) specifies the number of sample points along a path for which to return elevation data. The samples parameter divides the given path into an ordered set of equidistant points along the path.
- R Packages used:
  - library(**raster**)
  - library(**maptools**)
  - library(**rjson**)


DEVILS PEAK . TABLE MOUNTAIN . LIONS HEAD
BATRUN
South African
HOME
ENTER
INFO
ROUTE
GALLERY
RESULTS
I am running Batrun Tonight at 7pm
CP 1
CP 2
CP 3
This is how i used R to create elevation contour lines preparing for my race..
Route Info
Route Description
BATRUN 2016 will be a marked with reflective flags
The event starts at the Gardens Tech Rugby Club at 19h00.
It heads up Deerpark via the Plattekilp Wash Houses to Tafelberg Road where it connects to the Saddle Path of Devil's Peak.
Up and down Devil's Peak to return to Tafelberg Road.
Run along Tafelberg Road to the base of Platteklip Gorge. (Aid Station- drinks and food)
The route continues to the top of Platteklip, along the southern path to Maclear's Beacon, and back down to the base of Platteklip again. (Aid Station- drinks and food)
Follow Tafelberg Road to the Kloofnek parking lot. Cross the traffic circle and follow the road to Lion's Head (no marshals, please cross the road carefully!).
Take the path on the left that spirals clockwise around Lion's Head to the top and back to the parking lot at Kloof Nek the way you came.
Runners continue to the finish along the trails below Tafelberg Rd and down again via the Platteklip Wash Houses.
Google Maps Route
Cutoff Times
There will be cut-offs along the route. If you miss one you will be redirected to the finish by the marshals along the marked route.

Sourcing Elevation data in R

```
library(rjson)
#### Note: f is a prepared vector of path strings like below across the area looked at
#  in decimal degrees
####    “ 'path'='36.578581,-118.291994|36.23998,-116.83171' ”

results = lapply(f,function(path) {
  ELEVATION_BASE_URL = 'http://maps.google.com/maps/api/elevation/json'
  elvtn_args1 = list('path'=path,'samples'=90)
  usrp =
    URLencode(paste(paste(names(elvtn_args1),elvtn_args1,sep="="),collapse="&"))
  url = paste(ELEVATION_BASE_URL , '?' , usrp,sep="")
  response = fromJSON(file=url)
  do.call(rbind,lapply(response$results,function(rs)
            {cbind(rs$location$lat,rs$location$lng,rs$elevation)}))
})

mountain = do.call(rbind,results)
colnames(mountain) = c("LAT","LNG","ELEVATION")
write.csv(mountain, file="mountain.csv") # CSV data from google API
```

Creating GIS data in R

```
# Setup an 'empty' raster, here via an extent object
derived from your data
e <- extent(mountain[,1:2])
# package: raster
r <- raster(e, ncol=90, nrow=90)
# Function 'fun' below get applied for when there are
# multiple points per cell,
# Transfers values associated with point type
# spatial data into raster cells.
dem <- rasterize(mountain[, 1:2], r,
                 mountain[,3], fun=mean)
g <- as(dem, 'SpatialGridDataFrame')
```

Creating GIS data in R

```
# Make image object for contourLines function
# package: sp
im <- as.image.SpatialGridDataFrame(g);
intv = 20 # Contour Isolines Altitude difference.
# package: grDevices
cl <- contourLines(im, levels = seq(0, 1100, intv))

# Convert back to SpatialLinesDataFrame
# package: maptools
SLDF <- ContourLines2SLDF(cl)
proj4string(SLDF) <- CRS("+proj=longlat
+datum=WGS84")
```

Creating KML vector data in R

```
## convert SLDF to KML ,package: maptools
out <- sapply(slot(SLDF, "lines"), function(x) {
  m <- unique(sapply(cl, function(x) x$level))
            [as.numeric(gsub("C_", "", slot(x, "ID")))]
  # make thicker lines at 500 and 1000 m Isolines
  kmlLine(x, name = paste0(m, "m-Isoline"), description="-",
     col = "#FCCD47",
     lwd = ifelse(m%%500 == 0, ifelse(m%%1000 == 0, 1.8, 1.3), 0.5))
})

# write KML, package: maptools
kmlFile <- file("mountain.kml", "w")
cat(kmlLine(kmlname = "ContouR Lines",
      kmldescription = "<i>20 m Isolines by Jacques Booysen</i>")$header,
          file = kmlFile, sep = "\n")
cat(unlist(out["style", ]), file = kmlFile, sep = "\n")
cat(unlist(out["content", ]), file = kmlFile, sep = "\n")
cat(kmlLine()$footer, file = kmlFile, sep = "\n")
close(kmlFile)
```

(R)Results: 50m Contour lines On Google Maps

THE END

- Questions?

LIVE DEMO: http://tinyurl.com/zovyoun