Berlin Gentrification Data Preparation — Property Value Data

Setting up environment

Loading necessary packages

```
library(automap)
library(tidyverse)
library(httr)
library(raster)
library(sf)
library(tmap)
library(rgdal)
library(geojsonsf)
library(dismo)
library(cowplot)
```

Creating functions to download data from FIS Broker (background setup)

```
get_X_Y_coordinates <- function(x) {
  sftype <- as.character(sf::st_geometry_type(x, by_geometry = FALSE))
  if(sftype == "POINT") {
    xy <- as.data.frame(sf::st_coordinates(x))
    dplyr::bind_cols(x, xy)
  } else {
    x
  }
}

sf_fisbroker <- function(url) {
  typenames <- basename(url)
  url <- httr::parse_url(url)
  url$query <- list(service = "wfs",
                    version = "2.0.0",
                    request = "GetFeature",
                    srsName = "EPSG:25833",
                    TYPENAMES = typenames)
  request <- httr::build_url(url)
  print(request)
  out <- sf::read_sf(request)
  out <- sf::st_transform(out, 3035)
  out <- get_X_Y_coordinates(out)
  out <- st_as_sf(as.data.frame(out))
  return(out)
}
```

Loading Data

Downloading from Berlin's open data portal using their API; subsetting and cleaning.

Downloading the data:

```
brw2004 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2004")
brw2003 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2003")
brw2002 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2002")
load(url("https://userpage.fu-berlin.de/soga/300/30100_data_sets/berlin_district.RData"))
berlin_base <- st_transform(berlin.sf, 3035)
```

Processing the data:

```
# functions to make processing Bodenrichtwert data easier
brw_processing <- function(x){
  # casts objects from SpatialPolygons to sf dataframe
  y <- x %>% st_as_sf() %>%
    st_transform(3035) %>%
    # filters out only residential areas,
    dplyr::filter(NUTZUNG == "W - Wohngebiet") %>%
     # only selects relevant columns
    dplyr::select(gml_id, BRW, geometry) %>%
     # removes empty geometries
    filter(!st_is_empty(.))
  # removes outliers from data beyond the .95 threshold on a normal distribution
  val <- y$BRW
  yUL <- val %>% quantile(.95, na.rm = TRUE)
  out1 <- y %>%
    dplyr::filter(BRW < yUL)
  # casts the geometry of the dataframe from polygons to points for easier interpolation
  out <- st_centroid(out1)
  return(out)
}

# this really should be with fewer lines using lapply();
# as such, under construction.
brw2004 <- brw_processing(brw2004)
brw2003 <- brw_processing(brw2003)
brw2002 <- brw_processing(brw2002)

brw2003comb <- rbind(brw2002, brw2003, brw2004) %>% as_Spatial()
```

Rasterizing Data

Once loaded and processed, we begin rasterizing this data. This process relies extremely heavily upon Hiljmans (2016), found at: https://rspatial.org/raster/analysis/4-interpolation.html, both in terms of methods of rasterization and the evaluation of each method. The steps are outlined below:

Evaluation Functions

Creating template raster

```
berlin_template <- raster(extent(berlin_base),
                          resolution = 424,
                          crs = st_crs(berlin_base)$proj4string)
```

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj
## = prefer_proj): Discarded datum Unknown based on GRS80 ellipsoid in Proj4
## definition
```

Method 1: Proximity Polygons

This is a very basic method of interpolation that I'll use to compare to more advanced methods.

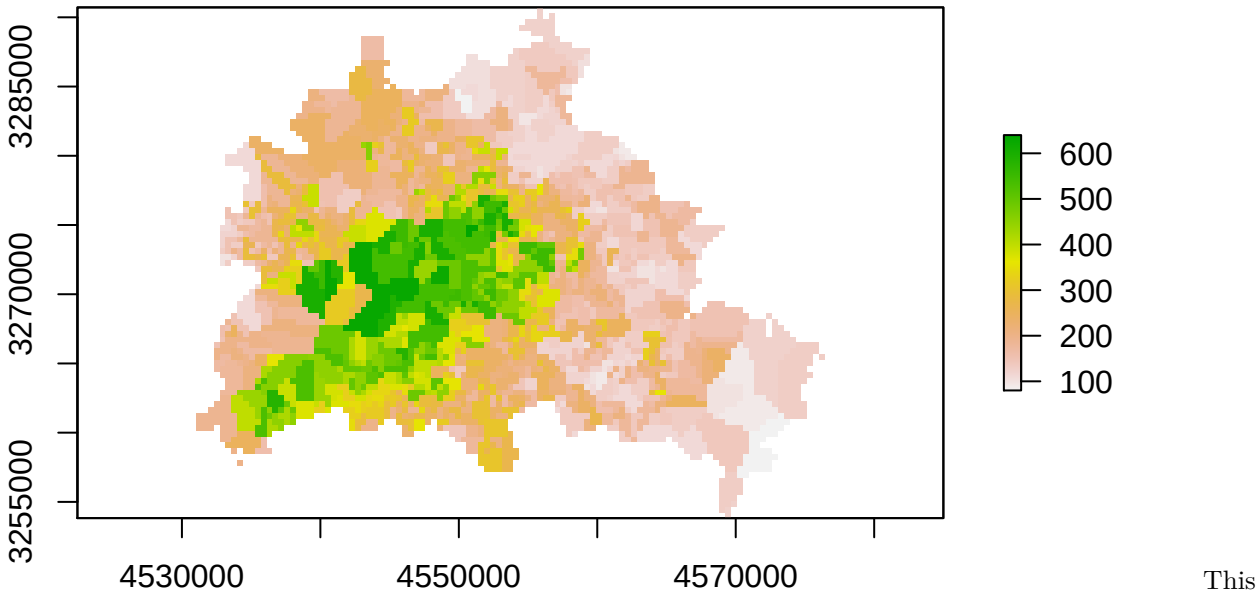
```
berlin_v <- voronoi(brw2003comb)
```

```
## Loading required namespace: deldir
```

```
vberlin <- raster::intersect(berlin_v, raster::aggregate(berlin_base))
```

```
## Loading required namespace: rgeos
```

```
b_pp <- rasterize(vberlin, berlin_template, "BRW")
plot(b_pp)
```



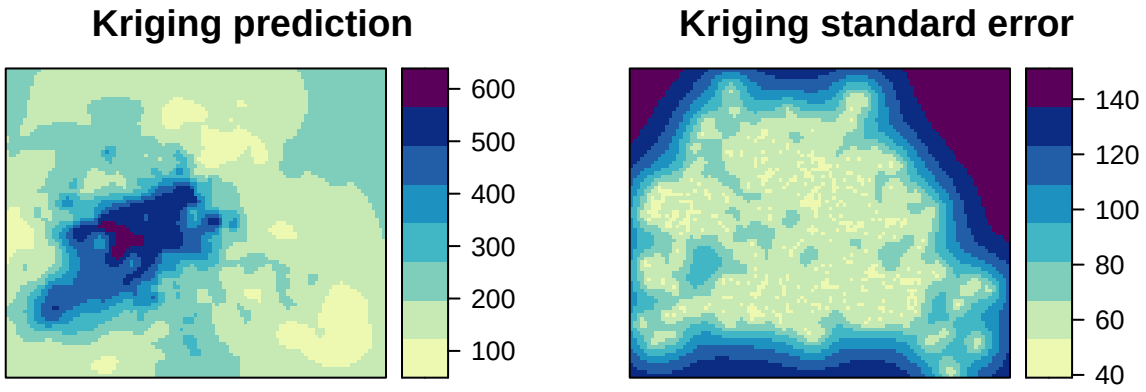
This method is just to provide an example of an extremely basic method of interpolation — we can see some really dramatic cutoffs around the city that might not do a great job of reflecting the actual price difference on different blocks. Nonetheless, still useful for comparison.

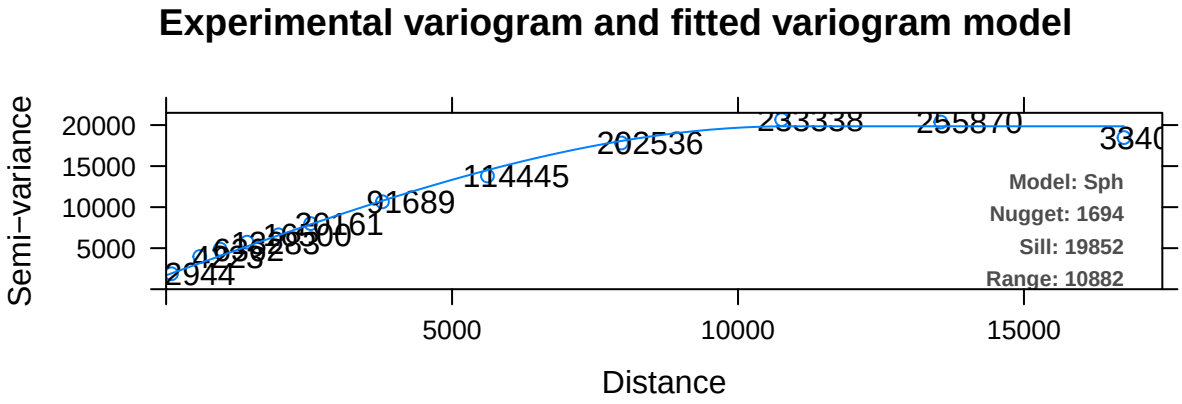
Method 2: Kriging

Kriging is a relatively complex method of interpolation — while the specifics of how exactly kriging works is beyond both my expertise in spatial statistics and the scope of this text, it is widely used for interpolation. We use the `automap` package to cut out a lot of the variogram calculation normally used in this context. This code automatically interpolates the parcel data, and compares each different possible variogram model, ultimately using an exponential model.

```
#unclear why this is necessary, but kriging won't complete unless the CRSs are reassigned.
bbase <- as(berlin_template, "SpatialPixels")
crs(brw2003comb) <- crs(bbase) <- "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS8
brw03_krigd <- autoKrige(formula=BRW~1, brw2003comb, new_data=bbase, model = c("Sph", "Exp", "Gau", "St
```

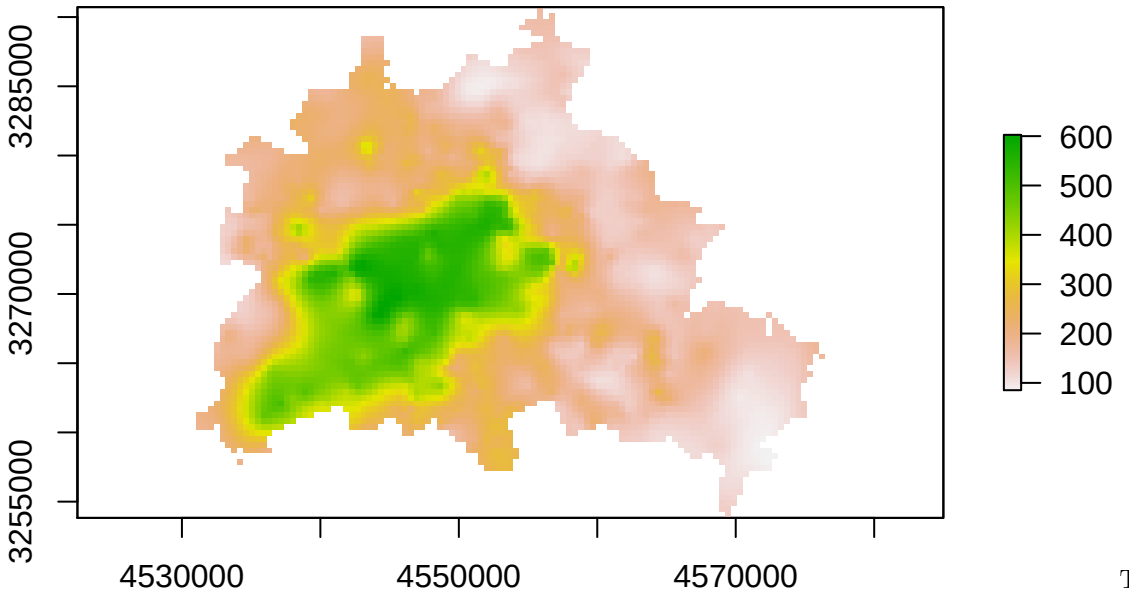
The `autoKrige()` function returns an object can show us quite a lot about the kriging interpolation, for anyone interested.





This is then converted to a raster

```
brw03_pred <- brw03_krigd$krige_output
brw03_pred <- raster(brw03_pred)
brw03_pred <- mask(x = brw03_pred, mask = berlin_base)
plot(brw03_pred)
```



This looks a lot better!

Next, we want to do this with all the `Bodenrichtwert` datasets from 2001-2019; the easiest way to do this is

by creating a function. This function will take the datasets, process them, and create the aggregated rasters similar to the other datasets.

```
brw_calc <- function(y1, y2, y3){
  df <- rbind(y1, y2, y3) %>% as_Spatial()
  crs(df) <- "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=m +no_defs"
  x <- autoKrige(formula=BRW~1, df, new_data=bbase, model = c("Sph", "Exp", "Gau", "Ste", "Nug"))
  y <- x$krige_output
  z <- raster(y)
  out <- mask(x = z, mask = berlin_base)
  return(out)
}
```

All of the Bodenrichtwert datasets are downloaded and run through this function to produce a raster for each odd year from 2003-2017.

Downloading the data:

```
brw2020 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2020") %>% brw_proc
brw2019 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2019") %>% brw_proc
brw2018 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2018") %>% brw_proc
brw2017 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2017") %>% brw_proc
brw2016 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2016") %>% brw_proc
brw2015 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2015") %>% brw_proc
brw2014 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2014") %>% brw_proc
brw2013 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2013") %>% brw_proc
brw2012 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2012") %>% brw_proc
brw2011 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2011") %>% brw_proc
brw2010 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2010") %>% brw_proc
brw2009 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2009") %>% brw_proc
brw2008 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2008") %>% brw_proc
brw2007 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2007") %>% brw_proc
brw2006 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2006") %>% brw_proc
brw2005 <- sf_fisbroker("https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_brw_2005") %>% brw_proc
```

```
## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
```

```
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS
## = dumpSRS, : Discarded datum European_Terrestrial_Reference_System_1989 in
## Proj4 definition: +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
## +ellps=GRS80 +units=m +no_defs
```

```
brw2003_krigd <- brw03_pred
brw2005_krigd <- brw_calc(brw2004, brw2005, brw2006)
brw2007_krigd <- brw_calc(brw2006, brw2007, brw2008)
brw2009_krigd <- brw_calc(brw2008, brw2009, brw2010)
brw2011_krigd <- brw_calc(brw2010, brw2011, brw2012)
brw2013_krigd <- brw_calc(brw2012, brw2013, brw2014)
brw2015_krigd <- brw_calc(brw2014, brw2015, brw2016)
brw2017_krigd <- brw_calc(brw2016, brw2017, brw2018)
brw2019_krigd <- brw_calc(brw2018, brw2019, brw2020)
```

Now that we have a heatmap of each year, lets take a look — note that the scales are different for each of

these, and real estate prices seem to increase quite dramatically after around 2009.

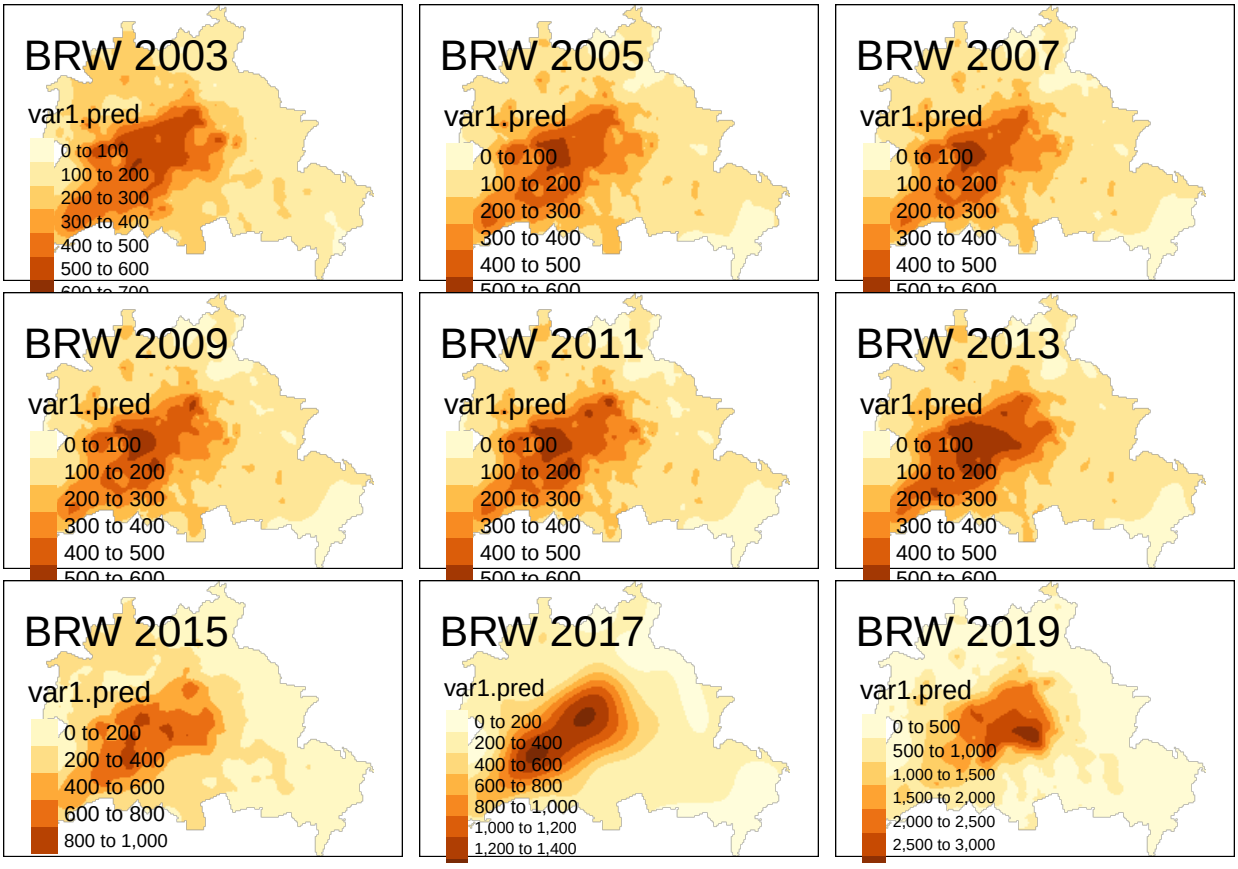
```
brw03plot <- qtm(brw2003_krigd, title="BRW 2003")
brw05plot <- qtm(brw2005_krigd, title="BRW 2005")
brw07plot <- qtm(brw2007_krigd, title="BRW 2007")
brw09plot <- qtm(brw2009_krigd, title="BRW 2009")
brw11plot <- qtm(brw2011_krigd, title="BRW 2011")
brw13plot <- qtm(brw2013_krigd, title="BRW 2013")
brw15plot <- qtm(brw2015_krigd, title="BRW 2015")
brw17plot <- qtm(brw2017_krigd, title="BRW 2017")
brw19plot <- qtm(brw2019_krigd, title="BRW 2019")
brw03thru19map <- tmap_arrange(brw03plot, brw05plot, brw07plot, brw09plot, brw11plot, brw13plot, brw15pl
brw03thru19map
```

```
## Some legend labels were too wide. These labels have been resized to 0.66, 0.66, 0.66, 0.66, 0.66. In
## Some legend labels were too wide. These labels have been resized to 0.66, 0.66, 0.66, 0.66, 0.66. In
## Some legend labels were too wide. These labels have been resized to 0.66, 0.66, 0.66, 0.66, 0.66. In
## Some legend labels were too wide. These labels have been resized to 0.66, 0.66, 0.66, 0.66, 0.66. In
## Some legend labels were too wide. These labels have been resized to 0.66, 0.66, 0.66, 0.66, 0.66. In

## Some legend labels were too wide. These labels have been resized to 0.66, 0.66, 0.66, 0.56. Increase

## Some legend labels were too wide. These labels have been resized to 0.52, 0.52, 0.52. Increase legend

## Some legend labels were too wide. These labels have been resized to 0.58, 0.51, 0.51, 0.51, 0.51, 0.5
```



Now, we can look at the chance in prices biyearly, and from 2003 to 2019.

```
brw05change <- brw2005_krigd - brw2003_krigd
brw07change <- brw2007_krigd - brw2005_krigd
brw09change <- brw2009_krigd - brw2007_krigd
brw11change <- brw2011_krigd - brw2009_krigd
brw13change <- brw2013_krigd - brw2011_krigd
brw15change <- brw2015_krigd - brw2013_krigd
brw17change <- brw2017_krigd - brw2015_krigd
brw19change <- brw2019_krigd - brw2017_krigd
brw03thru19change <- brw2019_krigd - brw2003_krigd
brwchangemap <- lapply(list(brw05change, brw07change, brw09change, brw11change, brw13change, brw15change
brwchangemap %>% tmap_arrange()
```