Climate futures for Haskell Indian Nations University

Install Climate Futures Toolbox

```
install.packages("cft")

if(!requireNamespace("tinytex", quietly = TRUE))
↪  install.packages("tinytex")

tinytex::install_tinytex()

# Install latest CRAN release
install.packages("papaja")

# Install remotes package if necessary
if(!requireNamespace("remotes", quietly = TRUE))
↪  install.packages("remotes")
```

Load other packages

```
library(tidyverse)
library(tidync)
library(cft)
library(sf)
library(ggplot2)
library(ggthemes)
library(ggpattern)
library(magick)
```

Finding basic layers in OpenStreetMap

1. Find the general area on Open Street Map We use a function from the osmdata package to find a bounding box for our area of interest. This is a nice function for this purpose because it can use plan language declarations (e.g. “Lawrence, Kansas” or “Boulder, Colorado”) for the location.

```
bb <- getbb("Lawrence, Kansas")
bb
```

```
##          min       max
## x -95.34454 -95.16662
## y  38.90447  39.03350
```

2. Find any buildings associated with any University in our “Lawrence, Kansas” bounding box. This request first calls the opq() function, which mounts to the OpenStreetMap database and then queries the “building” key (i.e. all the building footprints) for any building types matching the value “university”. This is a representation of the “key” and “value” system that OSM uses to query data. The final step is to convert the OSM output into a spatial format that works well in R, called sf.

```
my_boundary <- opq(bb) %>%
  add_osm_feature(key = "building", value = "university") %>%
osmdata_sf()

summary(my_boundary)
```

```
##                   Length Class  Mode
## bbox               1     -none- character
## overpass_call      1     -none- character
## meta               3     -none- list
## osm_points         5     sf     list
## osm_lines         40     sf     list
## osm_polygons      40     sf     list
## osm_multilines     0     -none- NULL
## osm_multipolygons 17     sf     list
```

3. The output from this request shows a list of multipolygons, polygons, linestrings, and points. Each of these data types have a different storage structure, so we can’t look at them all at the same time. Instead, lets start with polygons, which likely represent a single building footprint. Printing ‘my_boundary$osm_polygons’ shows that there are two Universities in Lawrence and we need to filter those results down to only include Haskell building.

```
boundaries <- my_boundary$osm_polygons %>%
  filter(operator == "Haskell Indian Nations University")
boundaries1 <- boundaries[1,] #take the first building (e.g. first row) of
↪  the returns
```

4. Plot our discovered footprint to visually confirm It looks like we found Winona Hall in the OpenStreetMap database. This is how we plot the perimeters associated with it.

```
basemap <- ggplot(data = boundaries1) +
  geom_sf(fill = "blueviolet") +
  geom_sf_text(aes(label = name), size=15) +
  theme_tufte()

basemap
```

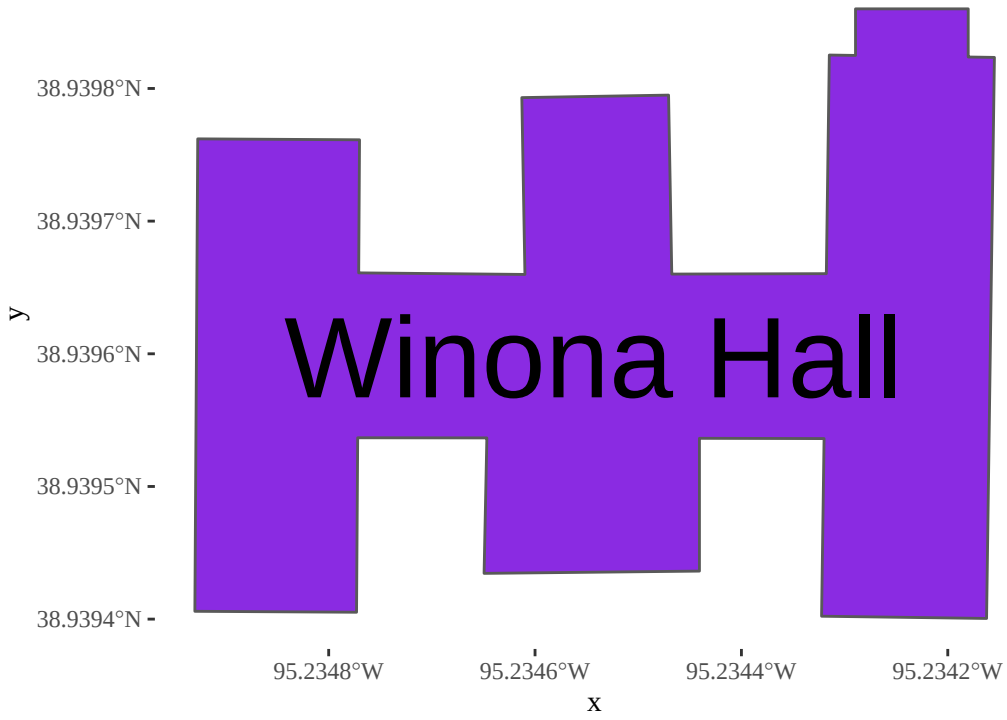


Figure 1: This seems to match.

Build a basemap from OpenStreetMap

Set your color palette

1. Download the Haskell university logo

```
seamless_image_filenames <- c(
  'Haskell_logo copy.jpg'
)
```

2. Sample the colors on that logo to make a custom color palette for our basemap

```
our_blue <- "#3A4E8B"
our_yellow <- "#FFD60F"
our_beige <- "#EDEDF0"
our_purple <- "#3E1471"
our_purple_alpha <- "#3E1471999"
```

Download all the layers you want to include in your basemap

1. Download the Haskell University footprint

```
my_boundary <- opq(bb) %>%
  add_osm_feature(key = "amenity", value = "university") %>%
osmdata_sf()

haskell_poly <- my_boundary$osm_multipolygons[1,]
```

2. Download street vector layers

```
# the big streets
streets <-
  opq(bb) %>%
  add_osm_feature(key = "highway",
                  value = c("motorway", "trunk",  "primary",
                            "secondary", "tertiary")) %>%
  osmdata_sf()

streets_crop <- streets$osm_lines %>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

small_streets <-
  opq(bb) %>%
  add_osm_feature(
    key = "highway",
    value = c("residential", "living", "unclassified",
              "service", "footway")) %>%
  osmdata_sf()

small_streets_crop <- small_streets$osm_lines %>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))
```

3. Download water layers

```
water <-
  opq(bb) %>%
  add_osm_feature(key = "waterway", value = "river") %>%
```

```
  osmdata_sf()

Kansas_river_multi <- water$osm_multilines %>%
  filter(name == "Kansas River")  %>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

Kansas_river_lines <- water$osm_lines %>%
  filter(name == "Kansas River")%>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

small_water_multi <- water$osm_multilines %>%
  filter(name != "Kansas River")%>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

small_water_lines <- water$osm_lines %>%
  filter(name != "Kansas River")%>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

reservoir <-
  opq(bb) %>%
  add_osm_feature(key = "water", value = "reservoir") %>%
  osmdata_sf()

reservoir_crop_multi <- reservoir$osm_multipolygons %>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

reservoir_crop_poly <- reservoir$osm_multipolygons %>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))

reservoir_crop <- cbind(reservoir_crop_multi, reservoir_crop_poly)

lake <-
  opq(bb) %>%
  add_osm_feature(key = "water", value = "lake") %>%
  osmdata_sf()
```

```
lake_crop <- lake$osm_polygons %>%
  st_crop(y = c(ymin = bb[2,1], ymax = bb[2,2], xmin = bb[1,1], xmax =
  ↪  bb[1,2]))
```

Combine layers into basemap

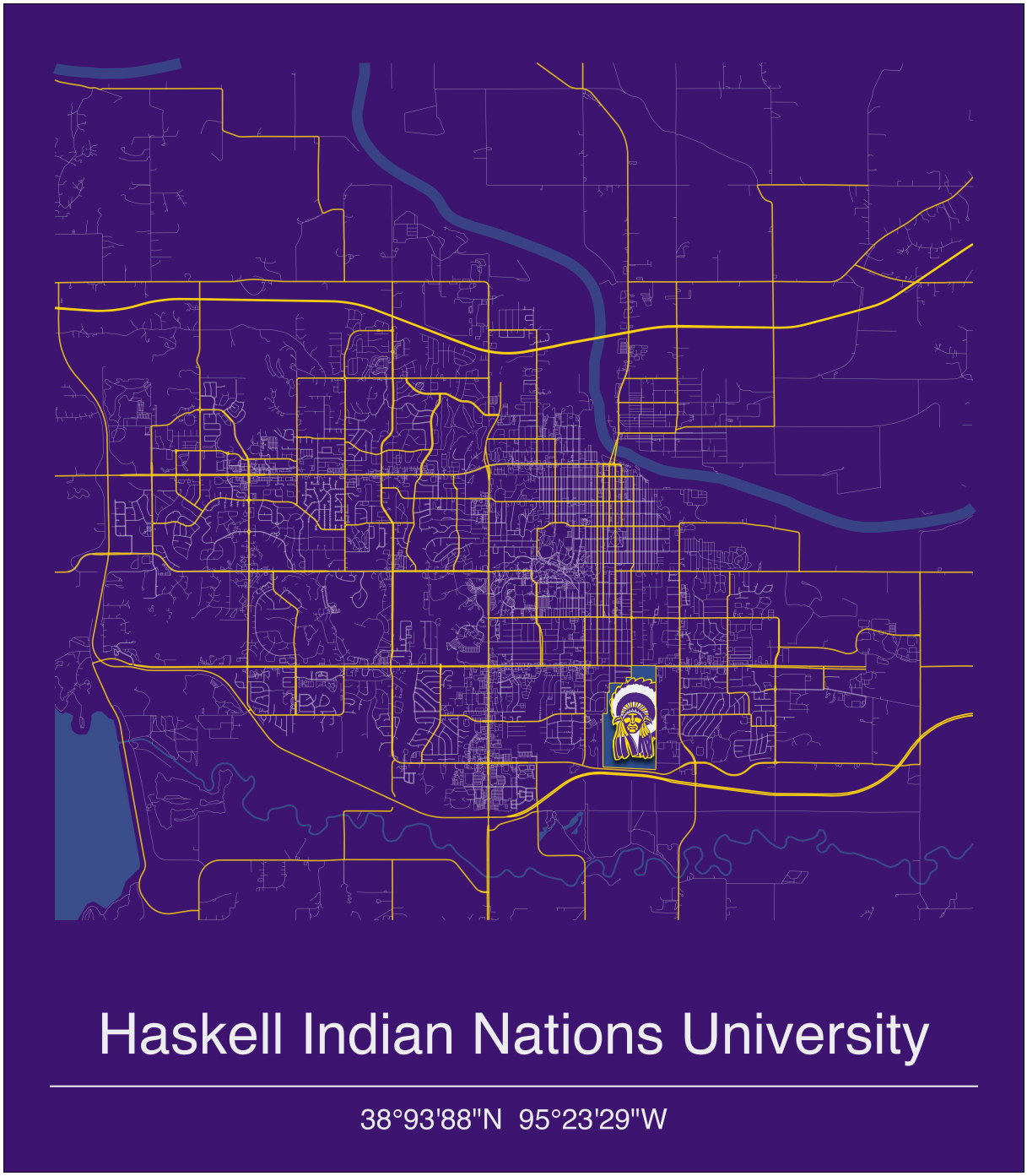
1. Adjust bounding box to be centered over Haskell University instead of Lawrence city center.

```
bbb <- bb
bbb[1,1] <- bbb[1,1] - 0.001
bbb[1,2] <- bbb[1,2] + 0.001
bbb[2,1] <- bbb[2,1] - 0.03
bbb[2,2] <- bbb[2,2] + 0.001
xlimit <- bbb[1,]
ylimit <- bbb[2,]
xmid <- xlimit[1] + diff(xlimit) / 2
ratio <- diff(xlimit) / diff(ylimit)
```

2. Stack the layers into a final basemap.

```
haskel_basemap <- ggplot() +
  # water
  geom_sf(data = Kansas_river_multi, alpha = .8,
          size = 3, colour = our_blue) +
  geom_sf(data = small_water_multi, alpha = .8,
          size = 0.5, colour = our_blue) +
  geom_sf(data = small_water_lines, alpha = .8,
          size = 0.5, colour = our_blue) +
  geom_sf(data = lake_crop, alpha = 1, fill = our_blue, size=0) +
  geom_sf(data = reservoir_crop, alpha = 1, fill = our_blue, size=0) +
  # small streets faded in the background
  geom_sf(data = small_streets_crop, alpha = .6,
          size = .1, colour = our_beige) +
  # large steets, making them stand out a bit with colour
  geom_sf(data = streets_crop, alpha = .8,
          size = .4, colour = our_yellow ) +
  # Haskell university property polygon
  geom_sf( data=haskell_poly, color=our_yellow, size=0.5) +
  # Haskell logo filling property polygon
```

```
   geom_sf_pattern(
     data = haskell_poly,
     size=0,
    pattern       = 'image',
    pattern_type  = 'tile',
    pattern_scale = 1.1,
    pattern_filename = seamless_image_filenames
  ) +
  # setting limits
  coord_sf(ylim = ylimit, xlim = xlimit, expand = TRUE) +
  # adding labels
  annotate(geom = "text", y = bbb[2,1]+ 0.013, x = xmid,
           label = "Haskell Indian Nations University", size = 12, colour =
           ↪  our_beige
           ) +
  annotate(geom = "errorbarh", xmin = xlimit[1], xmax = xlimit[2], y =
  ↪  bbb[2,1]+ 0.005, height = 0,
           size = 0.5, colour = our_beige) +
  annotate(geom = "text", y = bbb[2,1]+ 0.0001, x =  xmid,
           label = "38°93'88\"N  95°23'29\"W",  size = 6,
           colour = our_beige) +
  # finishing touches
  theme_void() +
  theme(panel.background = element_rect(fill = our_purple),
        plot.background = element_rect(fill = NA))

 #Note that the proportions will be off in this view. You should look at
 ↪  the png that was saved to see the correct view.

#haskel_basemap
```



3. Save the basemap in high resolution print

```
ggsave(haskel_basemap, filename = "All_roads_lead_to_Haskell.png", height =
↪  11, width=8.5, units="in", dpi=600)
```

Mount the climate dataset

This dataset is way too big to download to a particular machine. Instead, you mount to the analysis ready data cube (i.e. netCDF) and only download the subsetted data that you want to pull.

1. First we download a small sample of data to get the grid of coordinates available in the area.
2. Now we calculate the center point for measuring to the nearest climate data point.

```
haskel_centroid <- st_coordinates(st_centroid(haskell_poly))
lat_pt <- haskel_centroid[1,2]
lon_pt <- haskel_centroid[1,1]
```

6. Search through the database of climate prediction points to find which one is closest to our building.

```
web_link = "https://cida.usgs.gov/thredds/dodsC/macav2metdata_daily_future"

# Change to
↪  "https://cida.usgs.gov/thredds/catalog.html?dataset=cida.usgs.gov/macav2metdata_dail
↪  for historical data.

src <- tidync::tidync(web_link)
lons <- src %>% activate("D2") %>% hyper_tibble()
lats <- src %>% activate("D1") %>% hyper_tibble()
```

```
known_lon <- lons[which(abs(lons-lon_pt)==min(abs(lons-lon_pt))),]
known_lat <- lats[which(abs(lats-lat_pt)==min(abs(lats-lat_pt))),]

chosen_pt <- st_as_sf(cbind(known_lon,known_lat), coords = c("lon", "lat"),
↪  crs = "WGS84", agr = "constant")
```

```
l <- st_union(st_centroid(haskell_poly),st_as_sf(chosen_pt) ) %>%
  st_cast("LINESTRING")
```

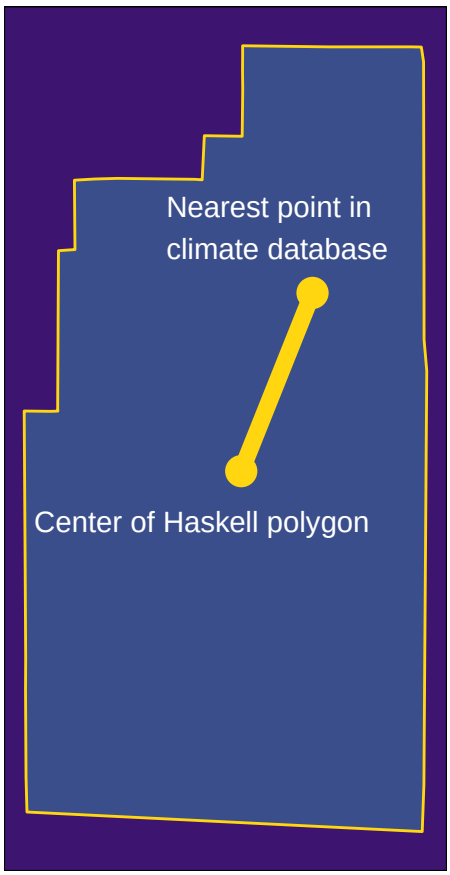
```
## Warning in st_centroid.sf(haskell_poly): st_centroid assumes attributes are
## constant over geometries of x

## Warning: attribute variables are assumed to be spatially constant throughout
## geometries
```

```
p <- st_union(st_centroid(haskell_poly),st_as_sf(chosen_pt) ) %>%
  st_cast("POINT") %>% mutate(label = c("Nearest point in \n climate
   ↪  database","Center of Haskell polygon"))
```

```
## Warning in st_centroid.sf(haskell_poly): st_centroid assumes attributes are
## constant over geometries of x

## Warning in st_centroid.sf(haskell_poly): attribute variables are assumed to b
## spatially constant throughout all geometries

## Warning in st_cast.sf(., "POINT"): repeating attributes for all sub-
geometries
## for which they may not be constant
```

```
box <- st_bbox(haskell_poly)
```

```
ggplot() +
   geom_sf( data=haskell_poly, color=our_yellow, size=0.5, fill=our_blue) +
  geom_sf(data= l, color=our_yellow, size=3) +
  geom_sf(data = chosen_pt, color=our_yellow, size=5) +
geom_sf(data = st_centroid(haskell_poly), color=our_yellow, size=5) +
  theme_void() +
  theme(panel.background = element_rect(fill = our_purple),
        plot.background = element_rect(fill = NA)) +
      geom_sf_text(data = p, aes(label = label), colour = "white",
       ↪  nudge_y=c(0.0013,-0.001), nudge_x = -0.001)
```




```
inputs <- cft::available_data()
```

```
## Trying to connect to the USGS.gov API

## not a file:
## ' https://cida.usgs.gov/thredds/dodsC/macav2metdata_daily_future '
##
## ... attempting remote connection

## Connection succeeded.

## Reading results

## Converting into an R data.table
```

```
head(inputs[[1]])
```

```
## # A tibble: 6 x 9
##   Available variab~1 Varia~2 Units Model Model~3 Scena~4 Varia~5 Model~6 Scen
##   <chr>              <chr>   <chr> <chr> <chr>   <chr>   <chr>   <chr>   <chr
## 1 huss_BNU-ESM_r1i1~ Specif~ kg k~ Beij~ r1i1p1  RCP 4.5 huss    BNU-
ESM rcp45
## 2 huss_BNU-ESM_r1i1~ Specif~ kg k~ Beij~ r1i1p1  RCP 8.5 huss    BNU-
ESM rcp85
## 3 huss_CCSM4_r6i1p1~ Specif~ kg k~ Comm~ r6i1p1  RCP 4.5 huss    CCSM4   rcp4
## 4 huss_CCSM4_r6i1p1~ Specif~ kg k~ Comm~ r6i1p1  RCP 8.5 huss    CCSM4   rcp8
## 5 huss_CNRM-CM5_r1i~ Specif~ kg k~ Cent~ r1i1p1  RCP 4.5 huss    CNRM-
C~ rcp45
## 6 huss_CNRM-CM5_r1i~ Specif~ kg k~ Cent~ r1i1p1  RCP 8.5 huss    CNRM-
C~ rcp85
## # ... with abbreviated variable names 1: `Available variable`, 2: Variable,
## #   3: `Model ensemble type (only CCSM4 relevant)`, 4: Scenario,
## #   5: `Variable abbreviation`, 6: `Model abbreviation`,
## #   7: `Scenario abbreviation`
```

```
head(inputs[[2]])
```

```
##   Available times      dates
## 1           38716 2006-01-01
## 2           38717 2006-01-02
## 3           38718 2006-01-03
## 4           38719 2006-01-04
## 5           38720 2006-01-05
## 6           38721 2006-01-06
```

```
head(unique(inputs[[1]]$Variable))
```

```
## [1] "Specific Humidity"                    "Precipitation"
## [3] "Maximum Relative Humidity"            "Minimum Relative Humidity"
## [5] "Surface Downswelling Shortwave Flux" "Maximum Temperature"
```

```
input_variables <- inputs$variable_names %>%
  filter(Variable %in% c("Maximum Relative Humidity",
                         "Minimum Relative Humidity",
                         "Maximum Temperature",
                         "Minimum Temperature",
                         "Precipitation",
                         "Eastward Wind",
                         "Northward Wind")) %>%
  filter(Scenario %in% c( "RCP 8.5")) %>%
  filter(Model %in% c(
    "Beijing Climate Center - Climate System Model 1.1",
    "Beijing Normal University - Earth System Model",
    "Canadian Earth System Model 2",
     ↪
  "Centre National de Recherches Météorologiques - Climate Model 5",
   ↪
  "Commonwealth Scientific and Industrial Research Organisation - Mk3.6.0",
   ↪
  "Community Climate System Model 4",
   ↪
  "Geophysical Fluid Dynamics Laboratory - Earth System Model 2 Generalized
   ↪  Ocean Layer Dynamics",
  "Geophysical Fluid Dynamics Laboratory - Earth System Model 2 Modular
   ↪  Ocean",
  "Hadley Global Environment Model 2 - Climate Chemistry 365 (day) ",
   ↪
 "Hadley Global Environment Model 2 - Earth System 365 (day)",
 ↪
 "Institut Pierre Simon Laplace (IPSL) - Climate Model 5A - Low
 ↪  Resolution",
 "Institut Pierre Simon Laplace (IPSL) - Climate Model 5A - Medium
 ↪  Resolution",
 "Institut Pierre Simon Laplace (IPSL) - Climate Model 5B - Low
 ↪  Resolution",
 "Institute of Numerical Mathematics Climate Model 4",
 ↪
 "Meteorological Research Institute - Coupled Global Climate Model 3",
 ↪
 "Model for Interdisciplinary Research On Climate - Earth System Model",
 ↪
 "Model for Interdisciplinary Research On Climate - Earth System Model -
 ↪  Chemistry",
```

```
 "Model for Interdisciplinary Research On Climate 5",
 ↪
 "Norwegian Earth System Model 1 - Medium Resolution"  )) %>%

  pull("Available variable")

head(as.data.frame(input_variables))
```

```
##                  input_variables
## 1       pr_BNU-ESM_r1i1p1_rcp85
## 2         pr_CCSM4_r6i1p1_rcp85
## 3      pr_CNRM-CM5_r1i1p1_rcp85
## 4 pr_CSIRO-Mk3-6-0_r1i1p1_rcp85
## 5       pr_CanESM2_r1i1p1_rcp85
## 6    pr_GFDL-ESM2G_r1i1p1_rcp85
```

```
times <- inputs$available_times

start_time <- Sys.time()

Pulled_data_single_space_single_timepoint <- inputs$src %>%
  hyper_filter(lat = lat <= c(center_point[4]+0.05) & lat >=
   ↪  c(center_point[2]-0.05)) %>%
  hyper_filter(lon = lon <= c(center_point[3]+0.05) & lon >=
   ↪  c(center_point[1]-0.05)) %>%
  #hyper_filter(time = times$`Available times` ==  44558) %>%
  hyper_tibble(select_var = input_variables[1:38]) %>%
  st_as_sf(coords = c("lon", "lat"), crs = 4326, agr = "constant")

end_time <- Sys.time()
print(end_time - start_time)
```

```
#library(devtools)
#install_github("earthlab/cft")
library(future)
n_cores <- availableCores() - 1
plan(multisession, workers = n_cores)
```

```
out <- single_point_firehose(input_variables, known_lat, known_lon )
head(out)
```

```
haskell <- out
save(haskell, file = "haskell.RData")
```

```
load(file = "haskell.RData")
```

```
library("tidyverse")
df_precip <- haskell %>%
  st_drop_geometry() %>%
  select(time, "pr_BNU-ESM_r1i1p1_rcp85",
 "pr_CCSM4_r6i1p1_rcp85",
 "pr_CNRM-CM5_r1i1p1_rcp85",
 "pr_CSIRO-Mk3-6-0_r1i1p1_rcp85",      "pr_CanESM2_r1i1p1_rcp85",
  ↪
  "pr_GFDL-ESM2G_r1i1p1_rcp85",         "pr_GFDL-ESM2M_r1i1p1_rcp85",
   ↪
  "pr_IPSL-CM5A-MR_r1i1p1_rcp85",       "pr_IPSL-CM5B-LR_r1i1p1_rcp85",
   ↪
 "pr_MIROC-ESM-CHEM_r1i1p1_rcp85",     "pr_MIROC-ESM_r1i1p1_rcp85",
  ↪
 "pr_MIROC5_r1i1p1_rcp85",             "pr_NorESM1-M_r1i1p1_rcp85") %>%
  gather(key = "variable", value = "value", -time)
head(df_precip)
```

```
##    time                 variable value
## 1 38716 pr_BNU-ESM_r1i1p1_rcp85     0
## 2 38717 pr_BNU-ESM_r1i1p1_rcp85     0
## 3 38718 pr_BNU-ESM_r1i1p1_rcp85     0
## 4 38719 pr_BNU-ESM_r1i1p1_rcp85     0
## 5 38720 pr_BNU-ESM_r1i1p1_rcp85     0
## 6 38721 pr_BNU-ESM_r1i1p1_rcp85     0
```

```
df_min_temp <- haskell %>%
  st_drop_geometry() %>%
  select(time, unique(c("tasmax_BNU-ESM_r1i1p1_rcp85" ,
 "tasmax_CCSM4_r6i1p1_rcp85" ,         "tasmax_CNRM-CM5_r1i1p1_rcp85" ,
  ↪
```

```
 "tasmax_CSIRO-Mk3-6-0_r1i1p1_rcp85",  "tasmax_CanESM2_r1i1p1_rcp85"  ,
 ↪
 "tasmax_GFDL-ESM2G_r1i1p1_rcp85" ,    "tasmax_GFDL-ESM2M_r1i1p1_rcp85" ,
 ↪
 "tasmax_HadGEM2-CC365_r1i1p1_rcp85",  "tasmax_HadGEM2-ES365_r1i1p1_rcp85"
 ↪  ,
 "tasmax_IPSL-CM5A-LR_r1i1p1_rcp85",   "tasmax_IPSL-CM5A-MR_r1i1p1_rcp85"
 ↪  ,
 "tasmax_IPSL-CM5B-LR_r1i1p1_rcp85" ,
 ↪  "tasmax_MIROC-ESM-CHEM_r1i1p1_rcp85",
 "tasmax_MIROC-ESM_r1i1p1_rcp85",      "tasmax_MIROC5_r1i1p1_rcp85"  ,
 ↪
 "tasmax_MRI-CGCM3_r1i1p1_rcp85" ,     "tasmax_NorESM1-M_r1i1p1_rcp85",
 ↪
 "tasmax_inmcm4_r1i1p1_rcp85",         "tasmin_BNU-ESM_r1i1p1_rcp85" ,
 ↪
 "tasmin_CCSM4_r6i1p1_rcp85" ,         "tasmin_CNRM-CM5_r1i1p1_rcp85" ,
 ↪
 "tasmin_CSIRO-Mk3-6-0_r1i1p1_rcp85" , "tasmin_CanESM2_r1i1p1_rcp85" ,
 ↪
 "tasmin_GFDL-ESM2G_r1i1p1_rcp85" ,    "tasmin_GFDL-ESM2M_r1i1p1_rcp85" ,
 ↪
 "tasmin_HadGEM2-CC365_r1i1p1_rcp85",  "tasmin_HadGEM2-ES365_r1i1p1_rcp85"
 ↪  ,
 "tasmin_IPSL-CM5A-LR_r1i1p1_rcp85",   "tasmin_IPSL-CM5A-MR_r1i1p1_rcp85" ,
 ↪
 "tasmin_IPSL-CM5B-LR_r1i1p1_rcp85",
 ↪  "tasmin_MIROC-ESM-CHEM_r1i1p1_rcp85",
 "tasmin_MIROC-ESM_r1i1p1_rcp85",      "tasmin_MIROC5_r1i1p1_rcp85" ,
 ↪
 "tasmin_MRI-CGCM3_r1i1p1_rcp85",      "tasmin_NorESM1-M_r1i1p1_rcp85",
 ↪
 "tasmin_inmcm4_r1i1p1_rcp85",
"tasmax_bcc-csm1-1_r1i1p1_rcp85",
"tasmin_bcc-csm1-1_r1i1p1_rcp85"))) %>%
  gather(key = "variable", value = "value", -time)

df_min_temp <- df_min_temp[which(df_min_temp$value > 200), ]
df_min_temp$variable <-
 ↪  as.character(as.numeric(as.factor(df_min_temp$variable)))

head(df_min_temp)
```

```
##    time variable value
## 1 38716        2 280.2
## 2 38717        2 284.7
## 3 38718        2 285.6
## 4 38719        2 285.0
## 5 38720        2 288.1
## 6 38721        2 291.6
```

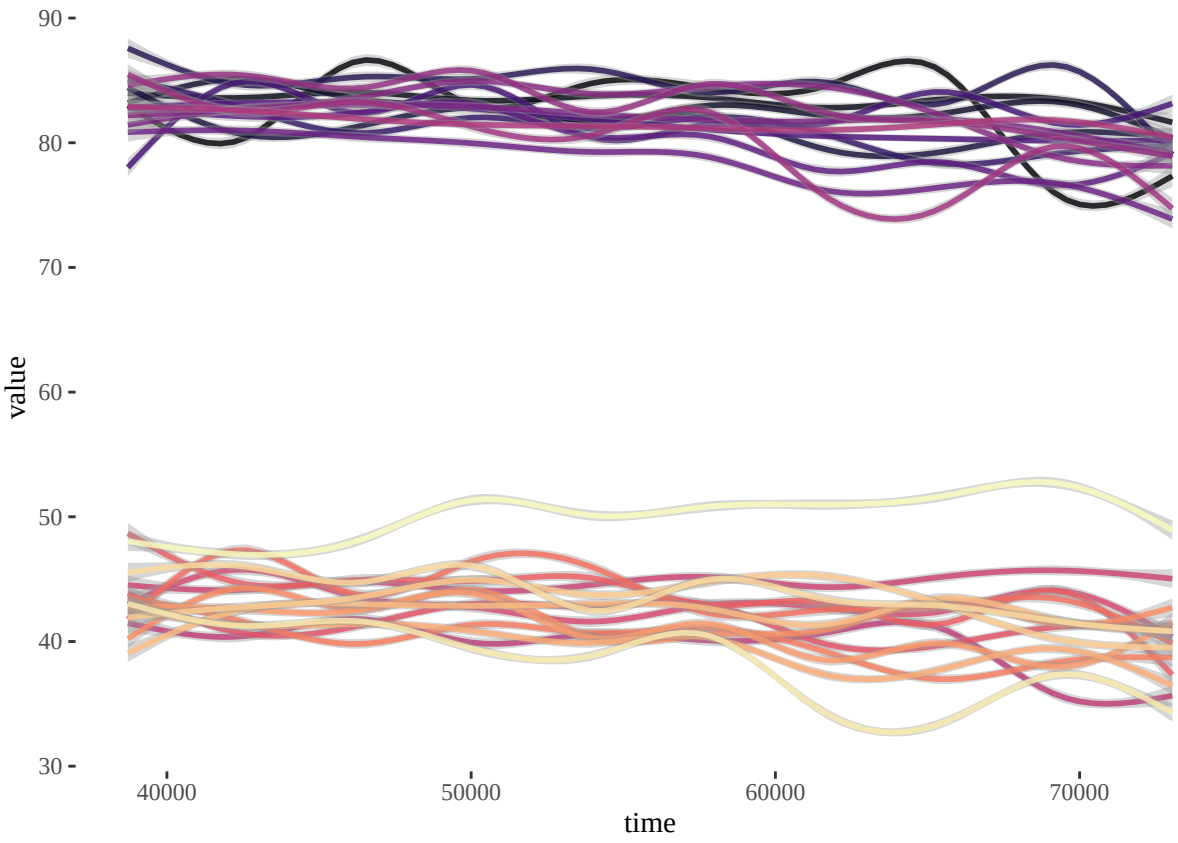
```
df_RH <- haskell %>%
  st_drop_geometry() %>%
  select(time,"rhsmax_BNU-ESM_r1i1p1_rcp85",
"rhsmax_CNRM-CM5_r1i1p1_rcp85",       "rhsmax_CSIRO-Mk3-6-0_r1i1p1_rcp85",
 "rhsmax_CanESM2_r1i1p1_rcp85",        "rhsmax_GFDL-ESM2G_r1i1p1_rcp85"  ,
 ↪
 "rhsmax_HadGEM2-CC365_r1i1p1_rcp85",  "rhsmax_HadGEM2-ES365_r1i1p1_rcp85"
 ↪  ,
 "rhsmax_IPSL-CM5A-LR_r1i1p1_rcp85",   "rhsmax_IPSL-CM5A-MR_r1i1p1_rcp85"
 ↪  ,
 "rhsmax_IPSL-CM5B-LR_r1i1p1_rcp85",
 ↪  "rhsmax_MIROC-ESM-CHEM_r1i1p1_rcp85",
 "rhsmax_MIROC-ESM_r1i1p1_rcp85" ,     "rhsmax_MIROC5_r1i1p1_rcp85"
 ↪  ,
 "rhsmax_MRI-CGCM3_r1i1p1_rcp85"  ,    "rhsmax_inmcm4_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_BNU-ESM_r1i1p1_rcp85"  ,      "rhsmin_CNRM-CM5_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_CSIRO-Mk3-6-0_r1i1p1_rcp85" , "rhsmin_CanESM2_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_GFDL-ESM2G_r1i1p1_rcp85"  ,   "rhsmin_GFDL-ESM2M_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_HadGEM2-CC365_r1i1p1_rcp85",  "rhsmin_HadGEM2-ES365_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_IPSL-CM5A-LR_r1i1p1_rcp85" ,  "rhsmin_IPSL-CM5A-MR_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_IPSL-CM5B-LR_r1i1p1_rcp85" ,
 ↪  "rhsmin_MIROC-ESM-CHEM_r1i1p1_rcp85",
 "rhsmin_MIROC-ESM_r1i1p1_rcp85"   ,   "rhsmin_MIROC5_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_MRI-CGCM3_r1i1p1_rcp85", ) %>%
  gather(key = "variable", value = "value", -time)

df_RH_min <- haskell %>%
```

```
  st_drop_geometry() %>%
  select(time,
 "rhsmin_BNU-ESM_r1i1p1_rcp85"  ,      "rhsmin_CNRM-CM5_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_CSIRO-Mk3-6-0_r1i1p1_rcp85" , "rhsmin_CanESM2_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_GFDL-ESM2G_r1i1p1_rcp85"  ,   "rhsmin_GFDL-ESM2M_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_HadGEM2-CC365_r1i1p1_rcp85",  "rhsmin_HadGEM2-ES365_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_IPSL-CM5A-LR_r1i1p1_rcp85" ,  "rhsmin_IPSL-CM5A-MR_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_IPSL-CM5B-LR_r1i1p1_rcp85" ,
 ↪  "rhsmin_MIROC-ESM-CHEM_r1i1p1_rcp85",
 "rhsmin_MIROC-ESM_r1i1p1_rcp85"   ,   "rhsmin_MIROC5_r1i1p1_rcp85"
 ↪  ,
 "rhsmin_MRI-CGCM3_r1i1p1_rcp85") %>%
  gather(key = "variable", value = "value", -time)

df_RH_max <- haskell %>%
  st_drop_geometry() %>%
  select(time,"rhsmax_BNU-ESM_r1i1p1_rcp85",
"rhsmax_CNRM-CM5_r1i1p1_rcp85",       "rhsmax_CSIRO-Mk3-6-0_r1i1p1_rcp85",
 "rhsmax_CanESM2_r1i1p1_rcp85",        "rhsmax_GFDL-ESM2G_r1i1p1_rcp85"  ,
 ↪
 "rhsmax_HadGEM2-CC365_r1i1p1_rcp85",  "rhsmax_HadGEM2-ES365_r1i1p1_rcp85"
 ↪  ,
 "rhsmax_IPSL-CM5A-LR_r1i1p1_rcp85",   "rhsmax_IPSL-CM5A-MR_r1i1p1_rcp85"
 ↪  ,
 "rhsmax_IPSL-CM5B-LR_r1i1p1_rcp85",
 ↪  "rhsmax_MIROC-ESM-CHEM_r1i1p1_rcp85",
 "rhsmax_MIROC-ESM_r1i1p1_rcp85" ,     "rhsmax_MIROC5_r1i1p1_rcp85"
 ↪  ,
 "rhsmax_MRI-CGCM3_r1i1p1_rcp85"  ,    "rhsmax_inmcm4_r1i1p1_rcp85"  ) %>%
  gather(key = "variable", value = "value", -time)
head(df_RH_max)
```

```
##    time                     variable value
## 1 38716 rhsmax_BNU-ESM_r1i1p1_rcp85   100
## 2 38717 rhsmax_BNU-ESM_r1i1p1_rcp85   100
## 3 38718 rhsmax_BNU-ESM_r1i1p1_rcp85   100
```

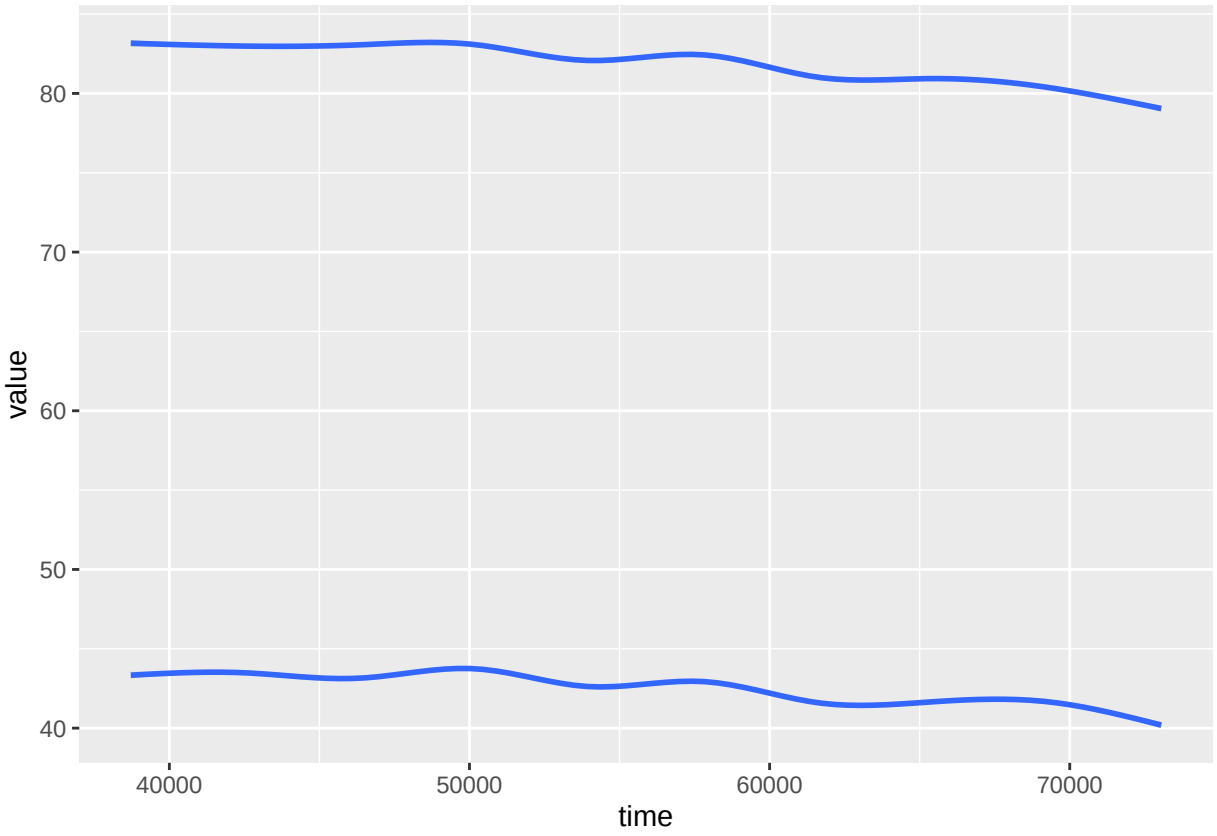
```
## 4 38719 rhsmax_BNU-ESM_r1i1p1_rcp85    95
## 5 38720 rhsmax_BNU-ESM_r1i1p1_rcp85   100
## 6 38721 rhsmax_BNU-ESM_r1i1p1_rcp85   100
```

```
all_climate_projections_RH <- ggplot(data= df_RH, aes(x = time, y = value,
↪  color = variable)) +
  #geom_line()+

  scale_colour_viridis_d(option="A", alpha = 0.8) +
  geom_smooth() +
  theme_tufte() +
  theme(legend.position = "none")


all_climate_projections_RH
```

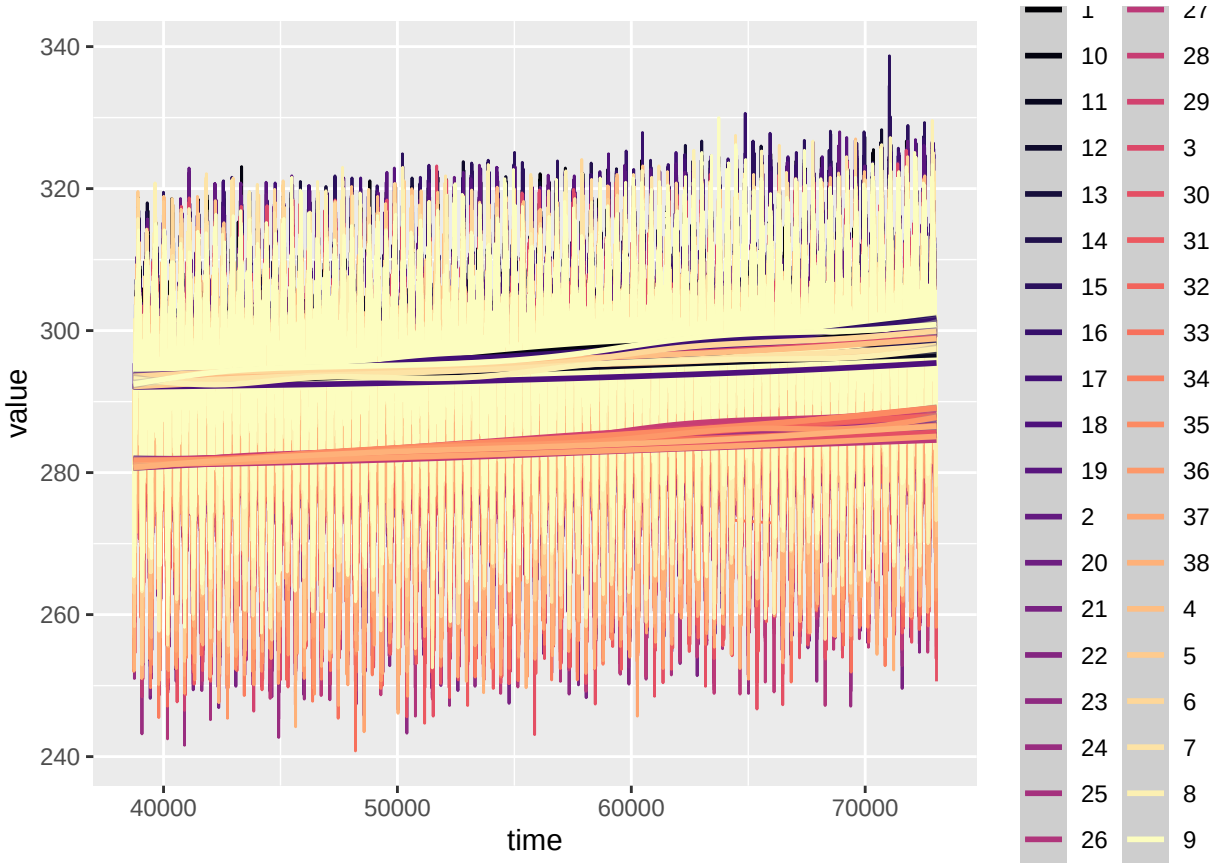



```
all_climate_projections_RH <- ggplot(data= df_RH_min, aes(x = time, y =
↪  value) )+
  geom_smooth(data= df_RH_min) +
```

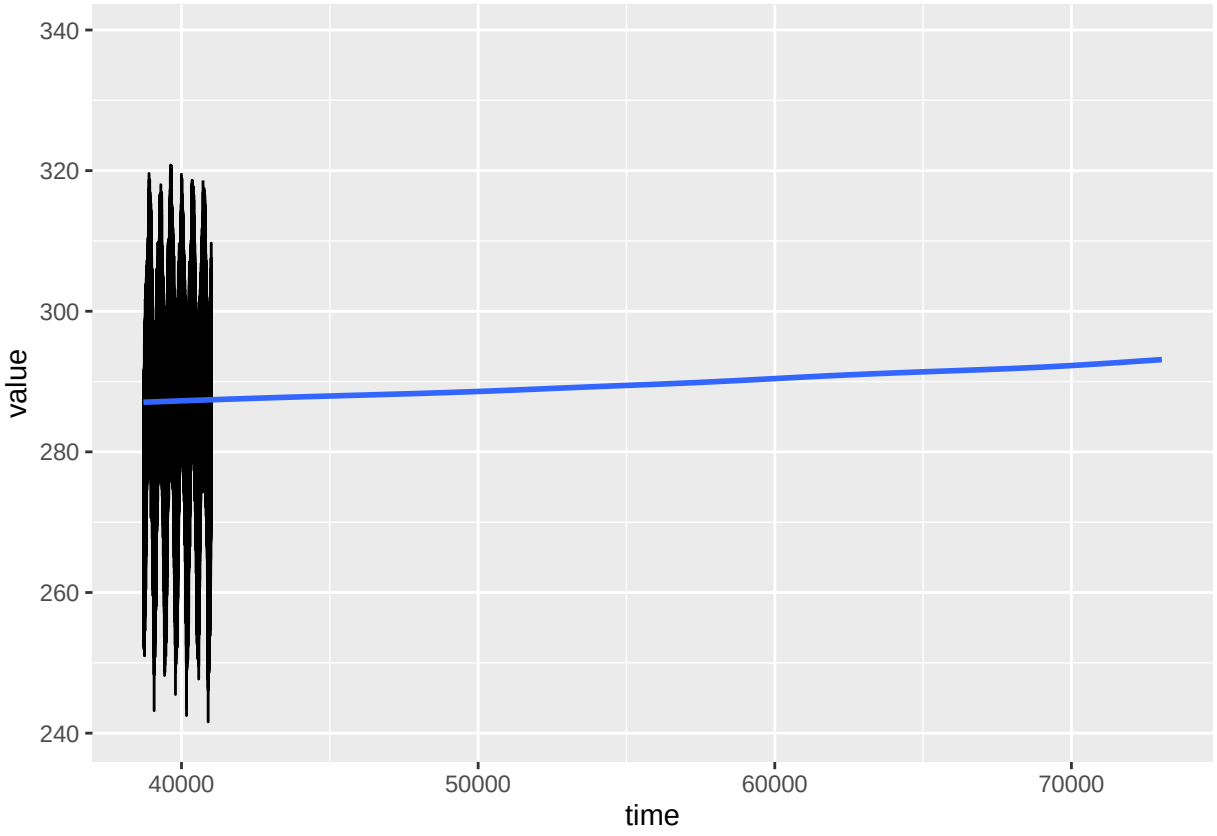
```
  geom_smooth(data= df_RH_max)


all_climate_projections_RH
```



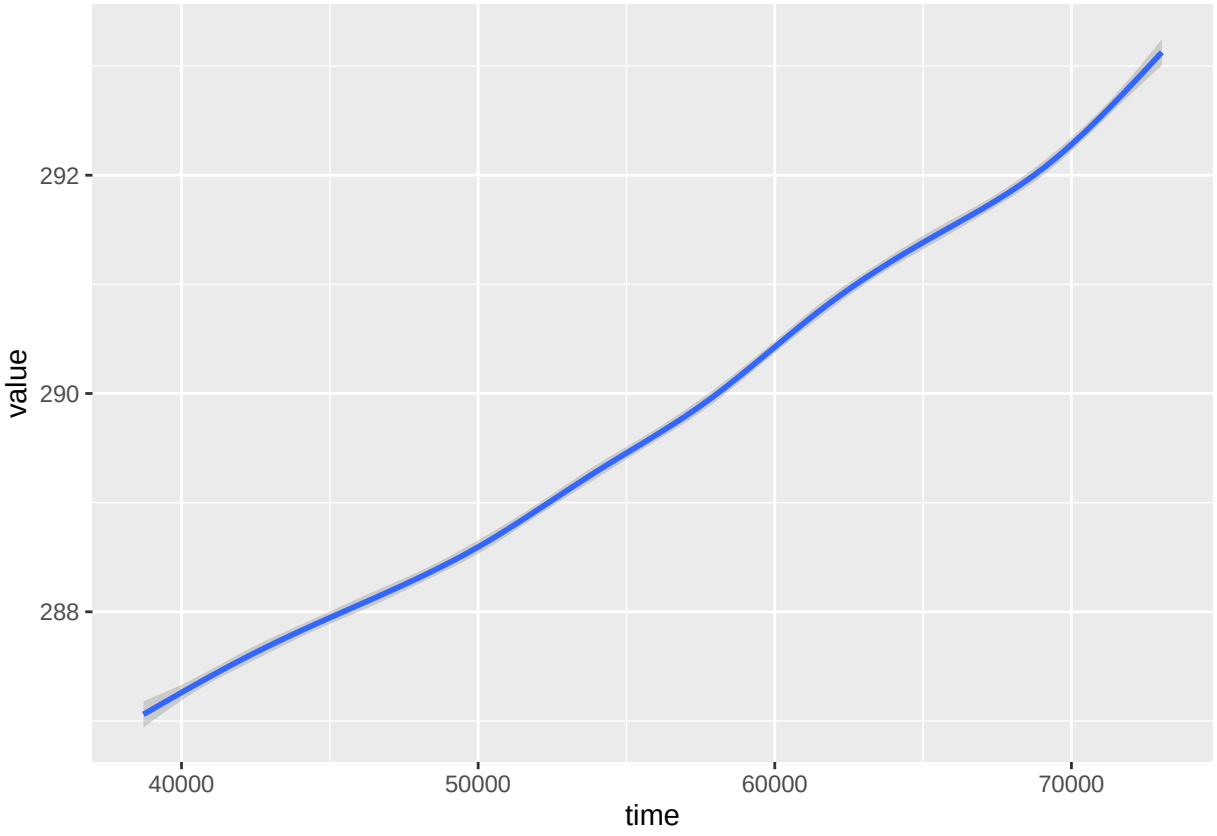

```
all_climate_projections <- ggplot(data= df_min_temp, aes(x = time, y =
↪  value, color = variable)) +
  geom_line()+
  geom_smooth() +
  scale_colour_viridis_d(option="A")


all_climate_projections
```



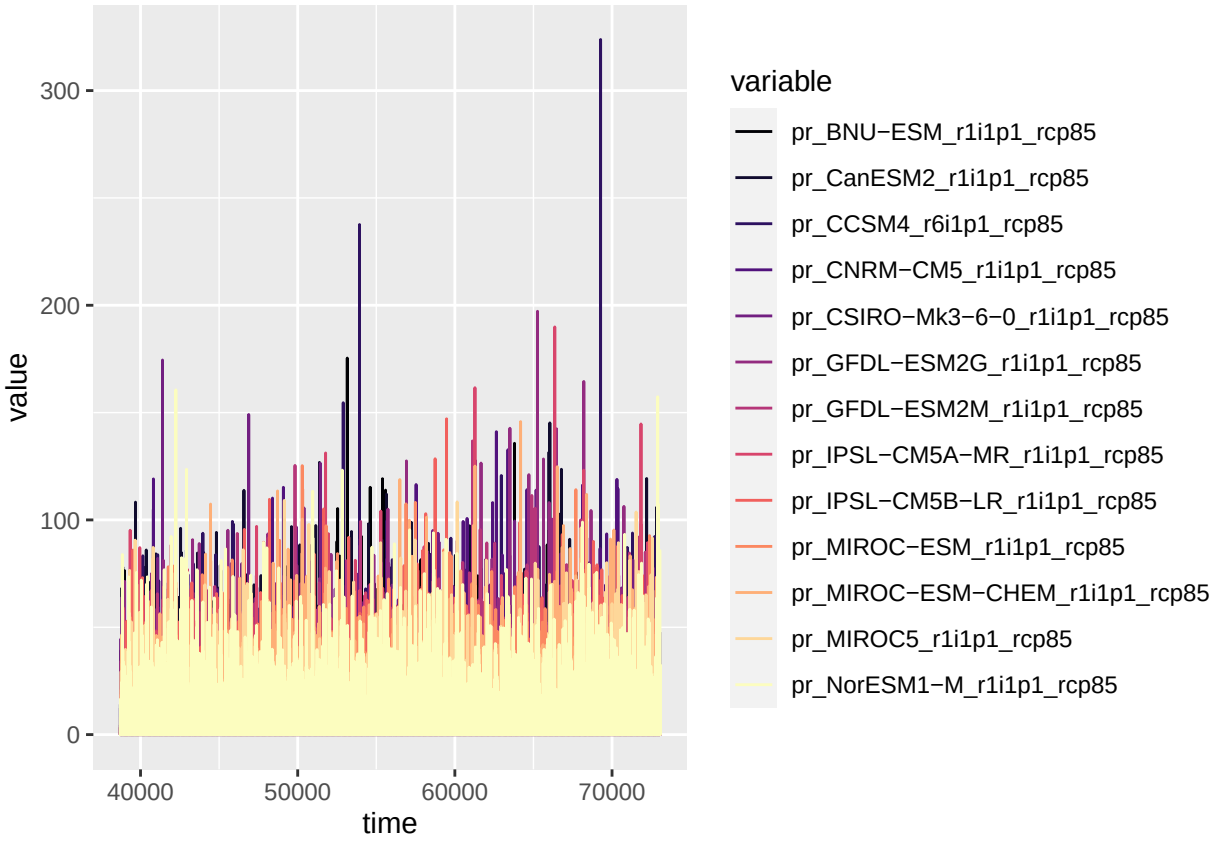

```
ensemble_climate_projections <- ggplot(data= df_min_temp, aes(x = time, y =
↪  value)) +
  geom_line()+
  geom_smooth() +
  scale_colour_viridis_d(option="A")


ensemble_climate_projections
```



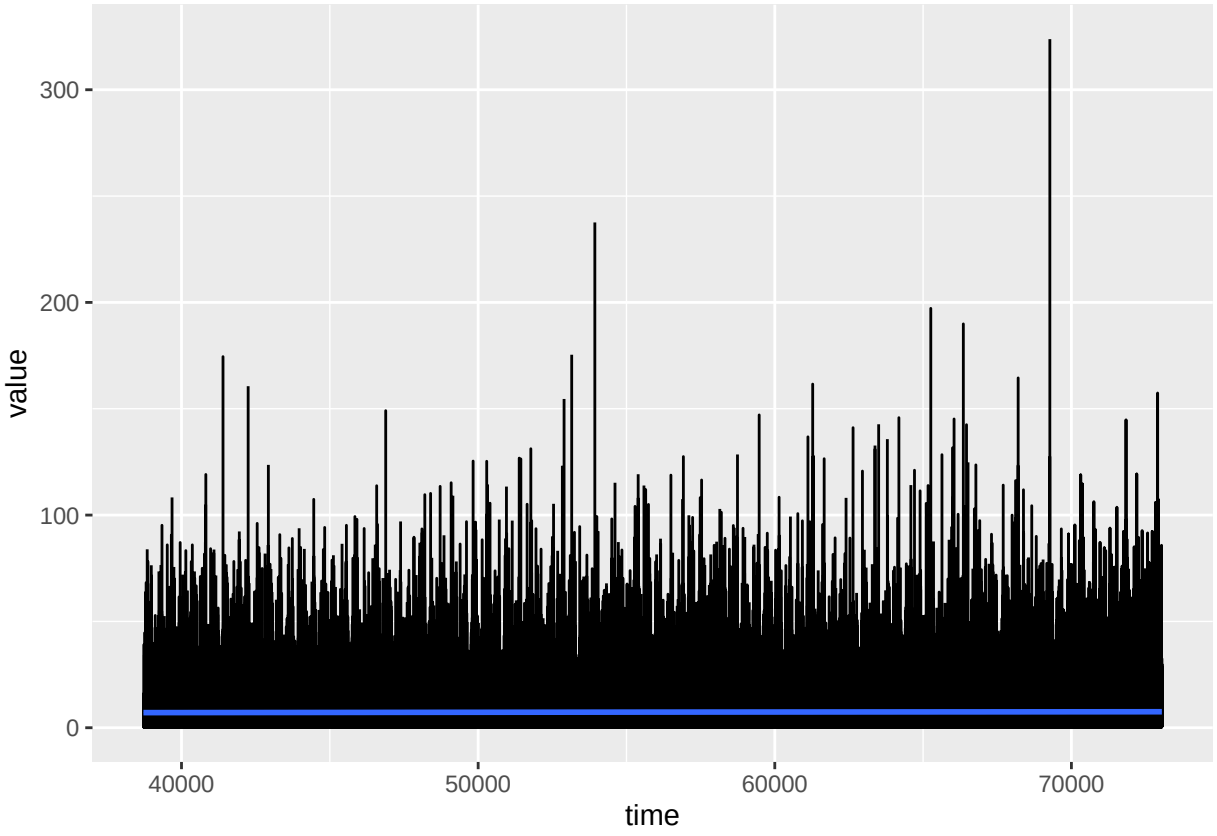

```
ggplot(data= df_min_temp, aes(x = time, y = value)) +
  geom_smooth() +
  scale_colour_viridis_c(option="A")
```



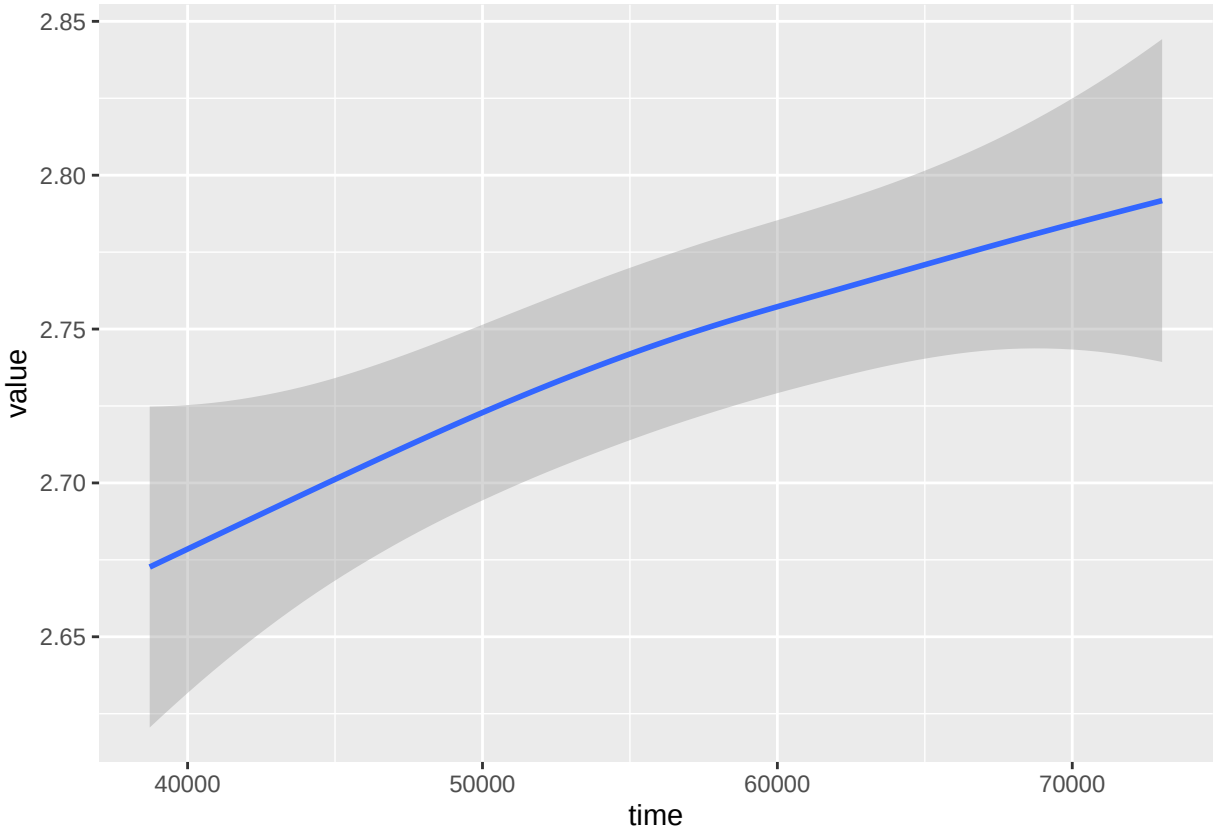

```
all_climate_projections <- ggplot(data= df_precip, aes(x = time, y = value,
↪  color = variable)) +
  geom_line()+
  scale_colour_viridis_d(option="A")


all_climate_projections
```




```
df_precip_2 <- df_precip[which(df_precip$value != 0),]

ensemble_climate_projections <- ggplot(data= df_precip_2, aes(x = time, y =
↪  value)) +
   geom_line()+
  geom_smooth() +
  scale_colour_viridis_d(option="A")


ensemble_climate_projections
```




```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth()
```

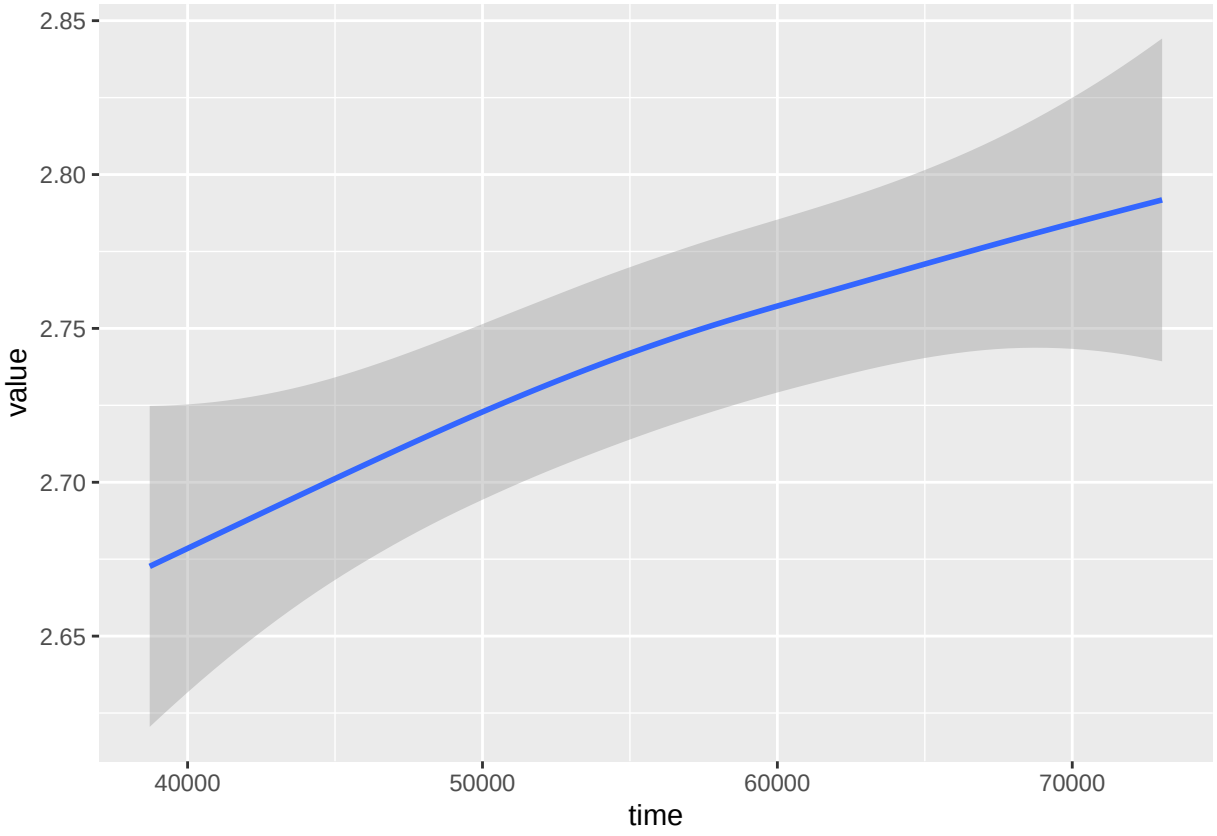



```
df_precip_quant_95 <- df_precip %>%
  mutate(quant = ntile(value, 100)) %>%
  filter(quant == 95)

df_precip_quant_96 <- df_precip %>%
  mutate(quant = ntile(value, 100)) %>%
  filter(quant == 96)

df_precip_quant_97 <- df_precip %>%
  mutate(quant = ntile(value, 100)) %>%
  filter(quant == 97)

df_precip_quant_98 <- df_precip %>%
  mutate(quant = ntile(value, 100)) %>%
  filter(quant == 98)

df_precip_quant_99 <- df_precip %>%
  mutate(quant = ntile(value, 100)) %>%
  filter(quant == 99)
```

```
df_precip_quant_100 <- df_precip %>%
  mutate(quant = ntile(value, 100)) %>%
  filter(quant == 100)

df_precip_quant_1000 <- df_precip %>%
  mutate(quant = ntile(value, 1000)) %>%
  filter(quant == 1000)

df_precip_quant_100000 <- df_precip %>%
  mutate(quant = ntile(value, 100000)) %>%
  filter(quant == 100000)
```

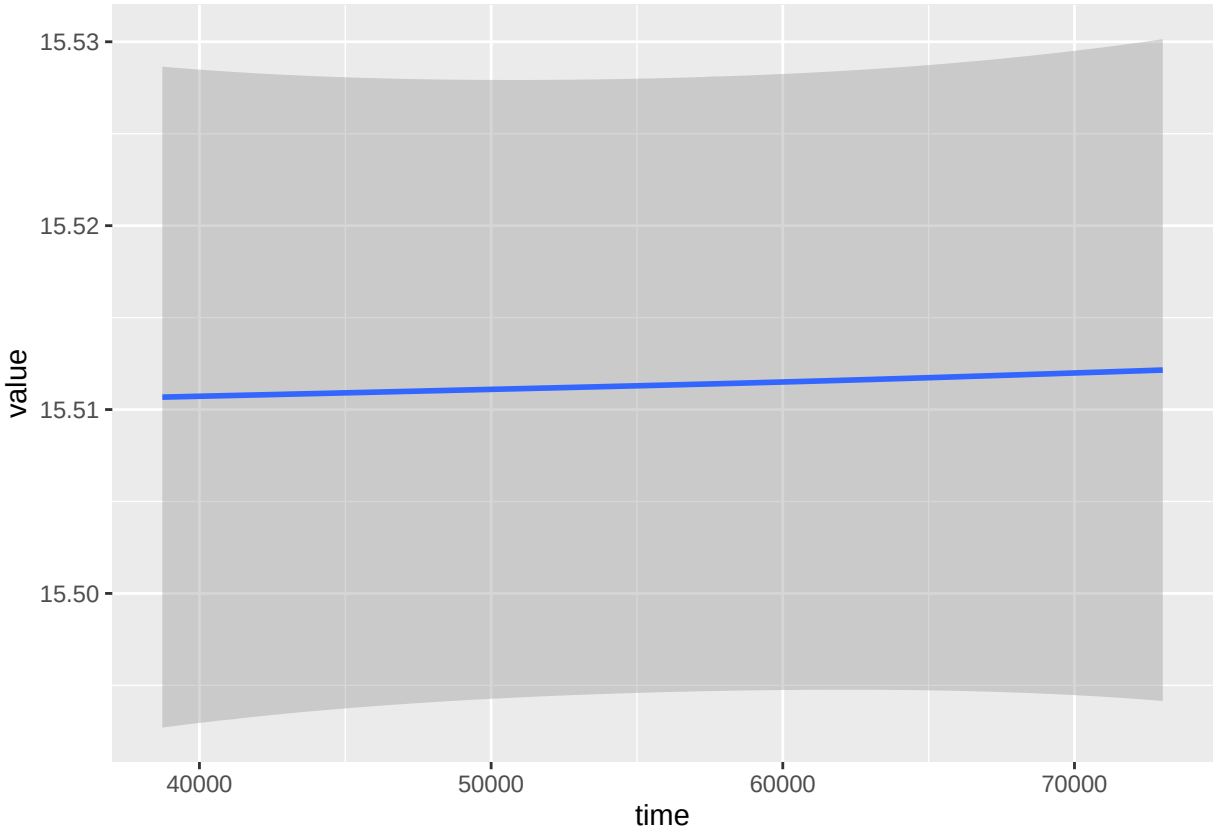
```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



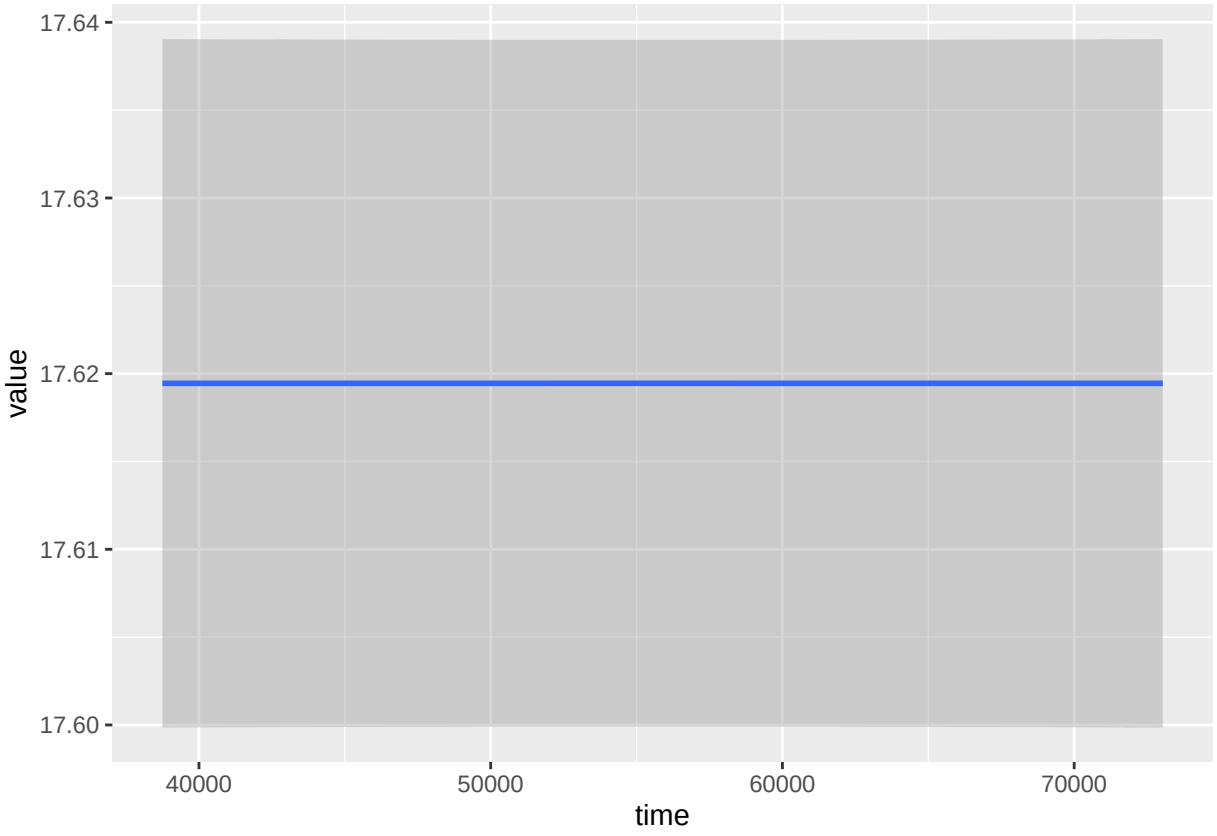
```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_95)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



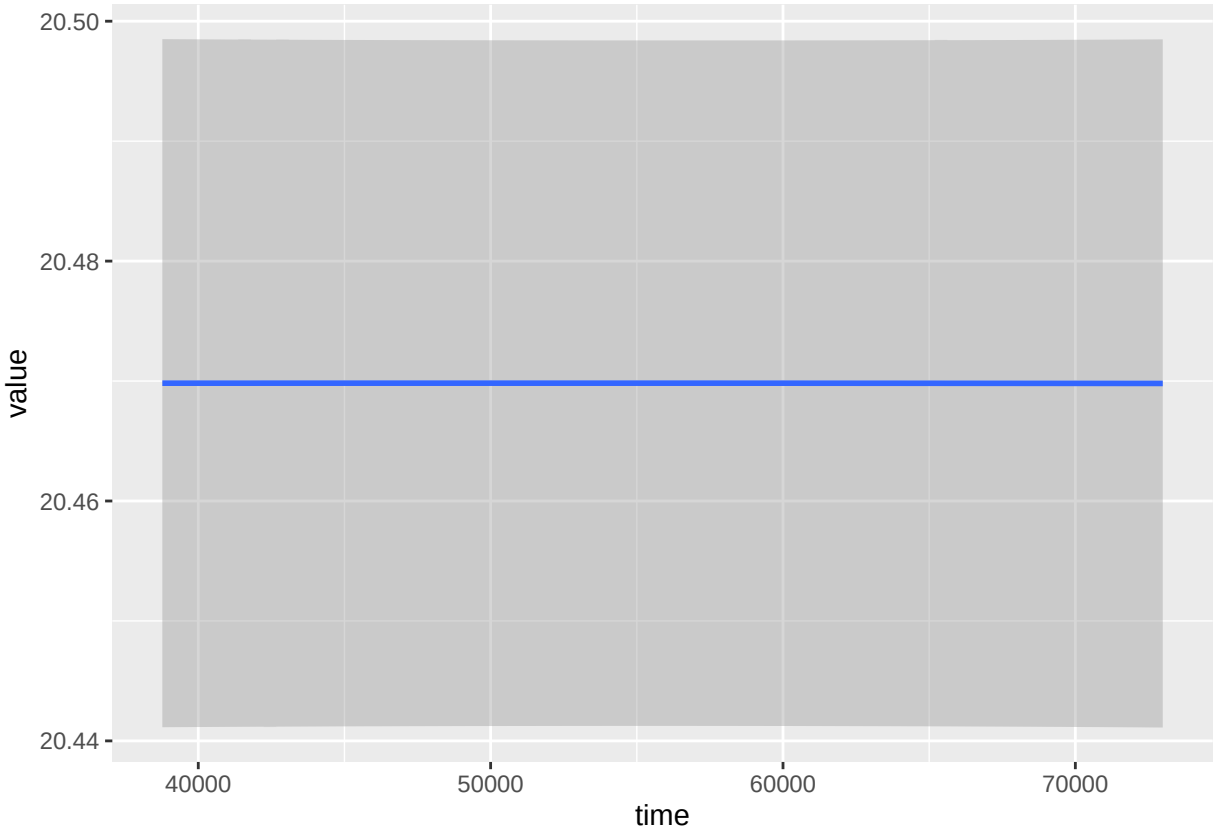

```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_96)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



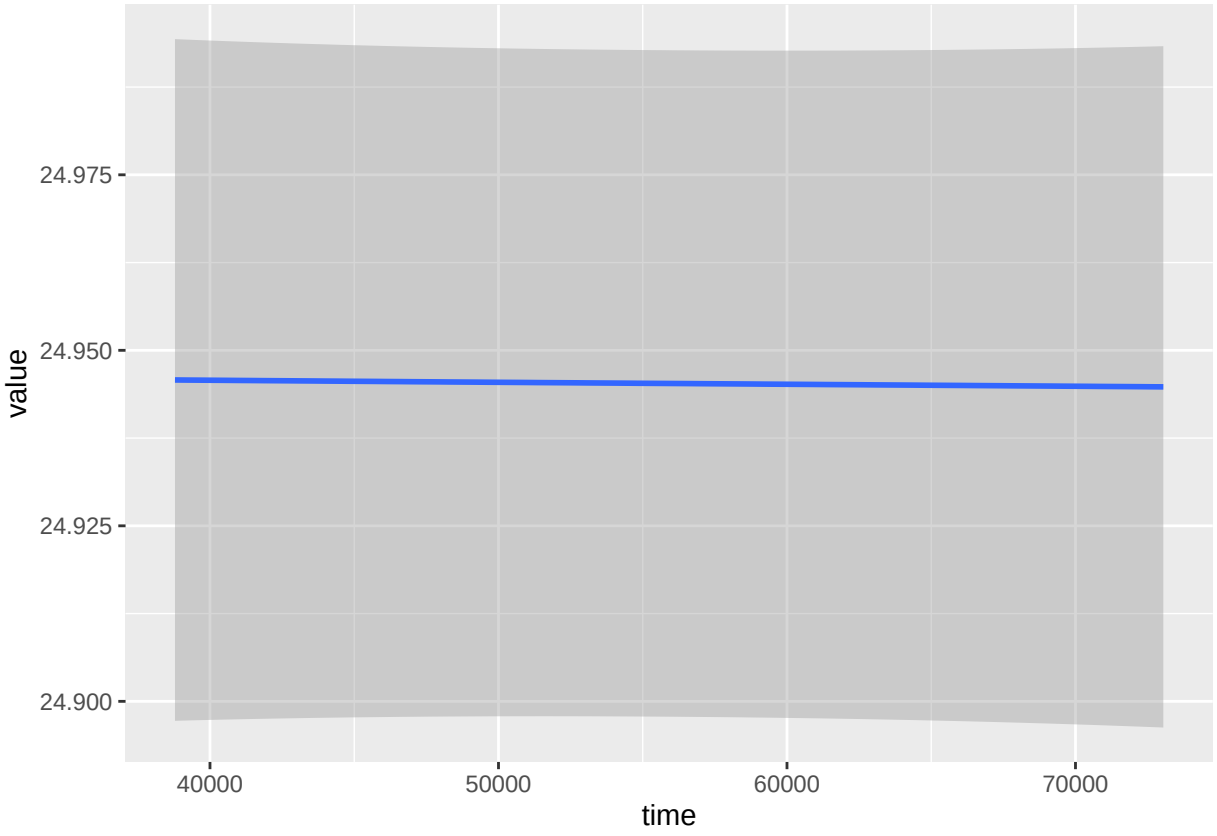

```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_97)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



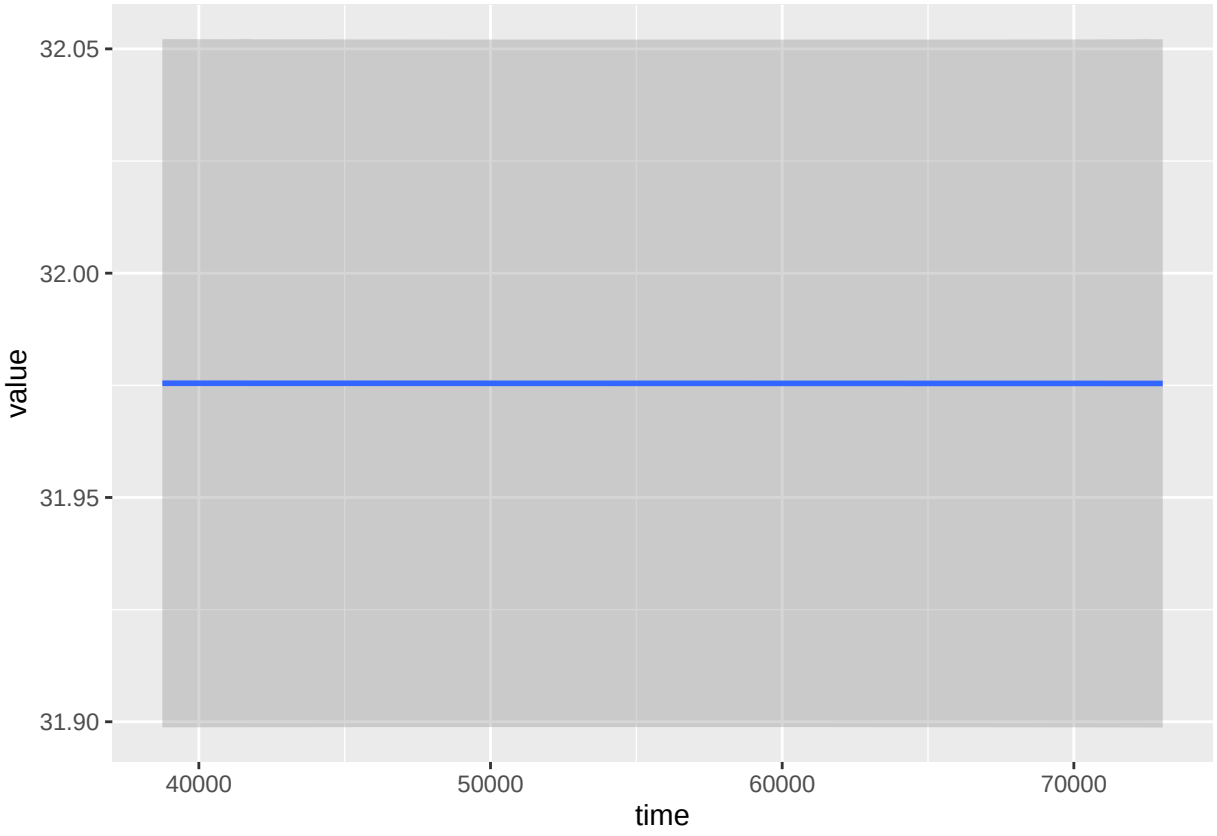

```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_98)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



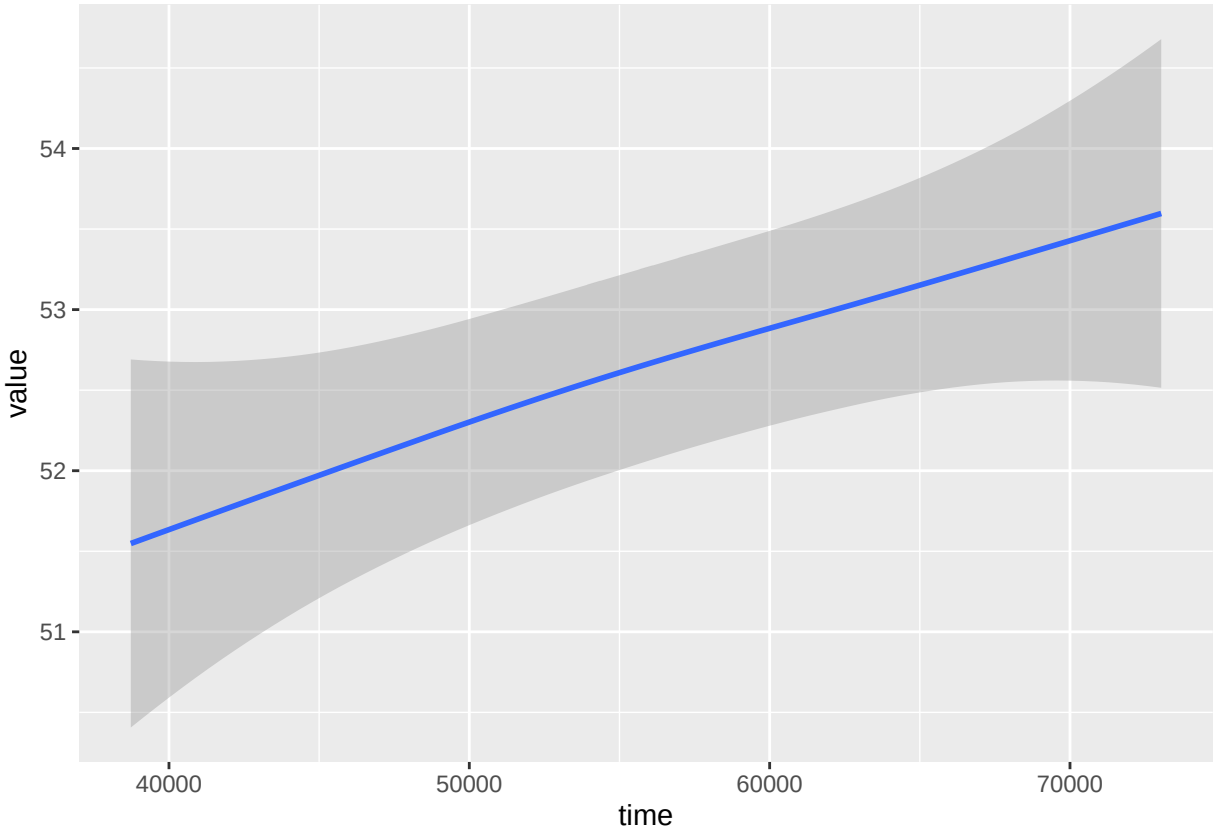

```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_99)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



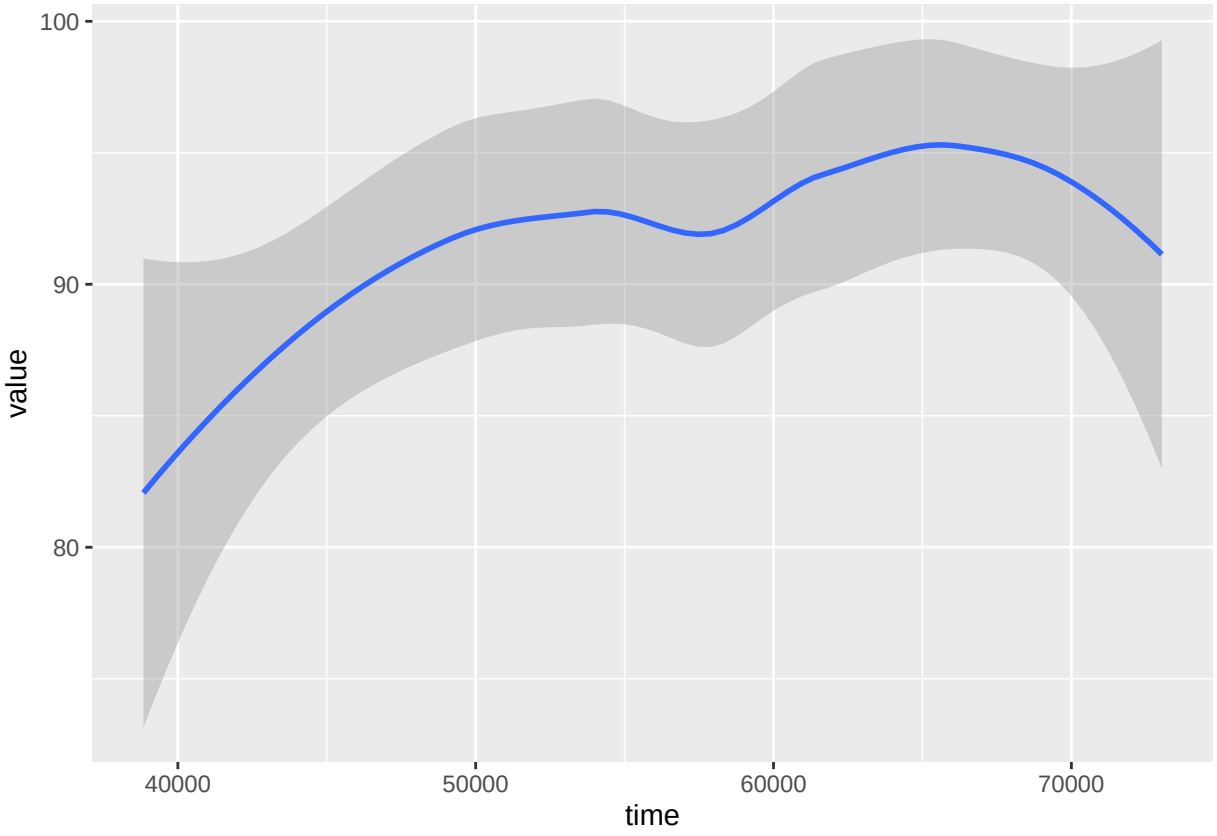

```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_100)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



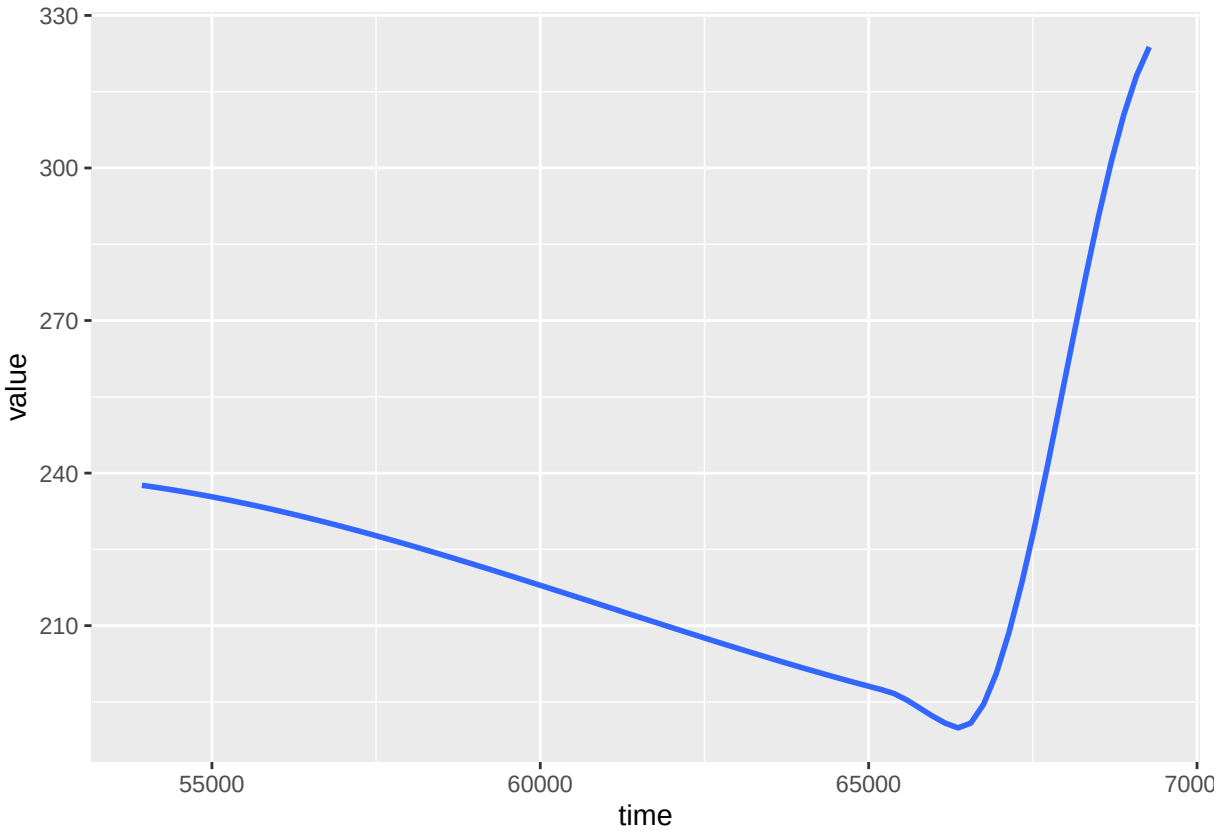

```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_1000)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip_quant_100000)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(data= df_precip, aes(x = time, y = value)) +
     geom_smooth(data= df_precip)+
     geom_smooth(data= df_precip_quant_95) +
    geom_smooth(data= df_precip_quant_96) +
    geom_smooth(data= df_precip_quant_97) +
    geom_smooth(data= df_precip_quant_98) +
    geom_smooth(data= df_precip_quant_99) +
  geom_smooth(data= df_precip_quant_100) +
  geom_smooth(data= df_precip_quant_1000) +
    geom_smooth(data= df_precip_quant_100000)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```