General outline

This code sample is intended to be a reproducible example of downloading, processing, and visualizing climate time series across the conterminous US. In this code we will explore a monthly time series of precipitation data from the GridMET data product (http://www.climatologylab.org/gridmet.html (http://www.climatologylab.org/gridmet.html)). We will detrend these precipitation data and transform these raw precipitation data into precipitation anomalies, which is just a simple deviation from the long term mean. By transforming into anomalies we can better understand how much increase of decrease precipitation is occurring across space and time, and how different those deviations are from the long term mean. At the very end we will extract the mean precipitation anomalies across defined regions in the conterminous US and create a series of time series plots.

First we need to setup our environment

```
packages <- c("tidyverse", "magrittr", "raster", "RCurl", "sf", "assertthat", 'lubrid
ate', 'ncdf4', 'snow', 'remote',
              'RColorBrewer', 'rasterVis', 'zoo', 'gridExtra', 'velox')
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  # automatically installs packages if not found
  install.packages(setdiff(packages, rownames(installed.packages())))
  # loads the library once installed
  lapply(packages, library, character.only = TRUE, quietly = TRUE)
} else {
  # if package is already install this loads it
  lapply(packages, library, character.only = TRUE, quietly = TRUE)
}
```

Now lets create some folders to store our raw and processed data in…

```
prefix <- "data"
raw_dir <- file.path(prefix, "raw")
us_dir <- file.path(raw_dir, "cb_2016_us_state_20m")
ecoregion_dir <- file.path(raw_dir, "na_cec_eco_l1")
climate_dir <- file.path(raw_dir, "climate")
processed_dir <- file.path(prefix, "processed")
processed_climate <- file.path(processed_dir, 'climate')

# Check if directory exists for all variable aggregate outputs, if not then create
var_dir <- list(prefix, raw_dir, us_dir, ecoregion_dir, climate_dir, processed_dir, p
rocessed_climate)
lapply(var_dir, function(x) if(!dir.exists(x)) dir.create(x, showWarnings = FALSE))
```

Let's define some small functions that we can call to make things more efficient and readable throughout the code.

```
# Create some simple functions to use later in the code
```

```r
# A function to download the data
download_data <- function(url, dir) {
  # because some files have a 'zip' extension while others dont we have to automatica
lly use the native extension
  file_extension <- url %>%
    basename %>%
    strsplit(., '[.]') %>%
    unlist
  file_extension_name <- file_extension[1] # get the file name
  file_extension_ext <- file_extension[2] # get the extension name

  download.file(url,
                destfile = file.path(dir, paste0(file_extension_name, ".", file_exten
sion_ext)))
  if(file_extension_ext == "zip"){
    unzip(file.path(dir, paste0(file_extension_name, ".", file_extension_ext)),
        exdir = dir)
  unlink(file.path(dir, paste0(file_extension_name, ".", file_extension_ext)))
  }
}

# A function to plot our raster data
raster_plot <- function(input_series, layout_vals, title) {
  rasterVis::levelplot(input_series,
                       layout = layout_vals,
                       par.settings = list(
                         axis.line = list(col = 'transparent')),
                       scales = list(draw = FALSE),
                       colorkey = TRUE, region = TRUE,
                       col.regions = colorRampPalette(brewer.pal(10, 'RdYlBu')),
                       xlab.top = title,
                       at=seq(min(minValue(input_series)), max(maxValue(input_series
)))) +
  layer(sp.polygons(as(study_area, 'Spatial'), lwd = 2)) # Overlay the usa states
}

# Creating anomalies
anom_fun <- function(x, y) {
  (x - y)
}

# Adding names to each layer in the raster stack
set_names <- function(x, start_date, end_date) {
  idx <- seq(as.Date(start_date), as.Date(end_date), by='month')
  var_names <- as.yearmon(idx)
  x <- setZ(x, var_names)
  names(x) <- as.character(var_names) # name the layers based on the month/year combi
nation
  return(x)
}
```

```
get_months <- function(start_date, end_date) {
  idx <- seq(as.Date(start_date), as.Date(end_date), by='month')
  var_names <- as.yearmon(idx)
  return(var_names)
}

# Get residuals to detrend our raw precipitation data
get_residuals <- function(x) {
  if (is.na(x[1])){
    rep(NA, length(x)) }
  else {
  m <- lm(x~time)
  q <- residuals(m) + predict(m)[1] # to remove the slope (detrend) add the residuals
and the intercept
  return(q)
  }
}

# A small function to classify our dates into seasons
classify_seasons <-  function(x) {
ifelse(x == 'Dec' | x == 'Jan' | x == 'Feb', "Winter",
       ifelse(x == 'Mar' | x == 'Apr' | x == 'May', "Spring",
              ifelse(x == 'Jun' | x == 'Jul' | x == 'Aug', "Summer",
                     "Fall")))
}

# Theme for plotting our time series in ggplot2
theme_pub <- function(base_size=11, base_family="") {
  library(grid)
  library(ggthemes)
  (theme_foundation(base_size=base_size, base_family=base_family)
    + theme(plot.title = element_text(hjust = 0.05, size = 13),

            panel.border = element_rect(colour = NA),
            panel.background = element_rect(colour = NA),
            panel.grid.major = element_line(colour="#f0f0f0"),
            panel.grid.minor = element_blank(),
            plot.background = element_rect(colour = NA),

            axis.line = element_line(colour="black"),
            axis.ticks = element_line(),

            legend.title = element_text(size=11),
            legend.position = "right",
            legend.text = element_text(size=11),
            legend.direction = "vertical",
            legend.key = element_rect(colour = "transparent", fill = "white"),
```

```
            strip.background=element_rect(colour=NA),
            strip.text.x = element_text(size = 10),

            axis.title = element_text(size = 11),
            axis.text.x = element_text(size = 10, angle = 65, hjust = 1),
            axis.text.y = element_text(size = 11)))
}

# A theme to plot vector maps through ggplot2
theme_map <- function(...) {
  theme_minimal(base_size=10) +
    theme(
      text = element_text(family = "", color = "#22211d"),
      axis.line = element_blank(),
      axis.text.x = element_blank(),
      axis.text.y = element_blank(),
      axis.ticks = element_blank(),
      axis.title.x = element_blank(),
      axis.title.y = element_blank(),

      panel.grid.major = element_line(color = "transparent", size = 0.2),
      panel.grid.minor = element_blank(),
      plot.background = element_blank(),
      panel.grid = element_blank(),
      panel.background = element_blank(),
      legend.background = element_blank(),
      panel.border = element_blank(),
      ...
    )
}
```

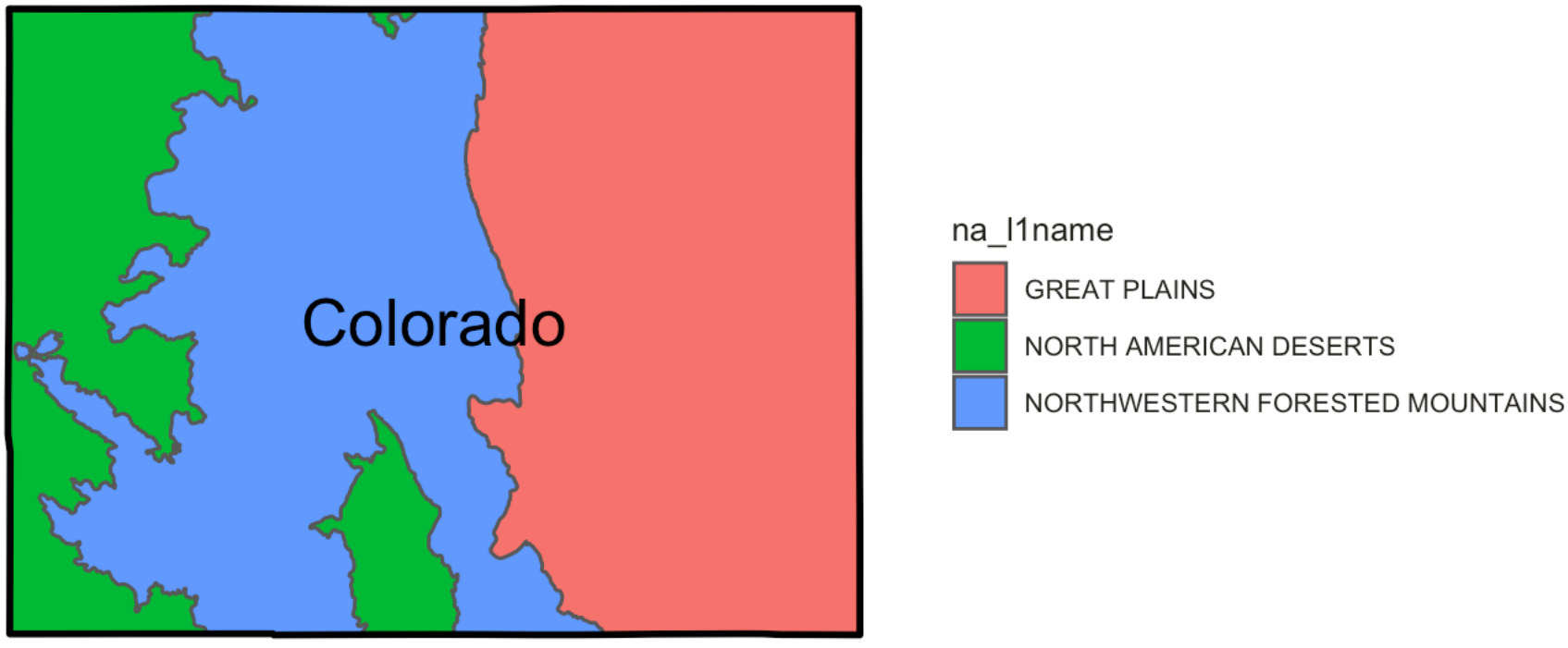
Now we can download the two data sets needed for this analysis

```
# USA states shapefile
# Make sure we haven't alredy downloaded the data
if(!file.exists(file.path(us_dir, "cb_2016_us_state_20m.shp"))) {
  download_data("https://www2.census.gov/geo/tiger/GENZ2016/shp/cb_2016_us_state_20m.
zip",
                us_dir)
}

# Level 1 ecoregions
# Make sure we haven't alredy downloaded the data
if(!file.exists(file.path(ecoregion_dir, "NA_CEC_Eco_Level1.shp"))) {
  download_data("ftp://newftp.epa.gov/EPADataCommons/ORD/Ecoregions/cec_na/na_cec_eco
_l1.zip",
                ecoregion_dir)
}

# Monthly Precipitation from 1979-2017
# Make sure we haven't alredy downloaded the data
if(!file.exists(file.path(climate_dir, 'pr_gridMET.nc'))) {
  download_data("https://www.northwestknowledge.net/metdata/data/monthly/pr_gridMET.n
c",
                climate_dir)
  }
```

Now that the data has been downloaded let's start processing!

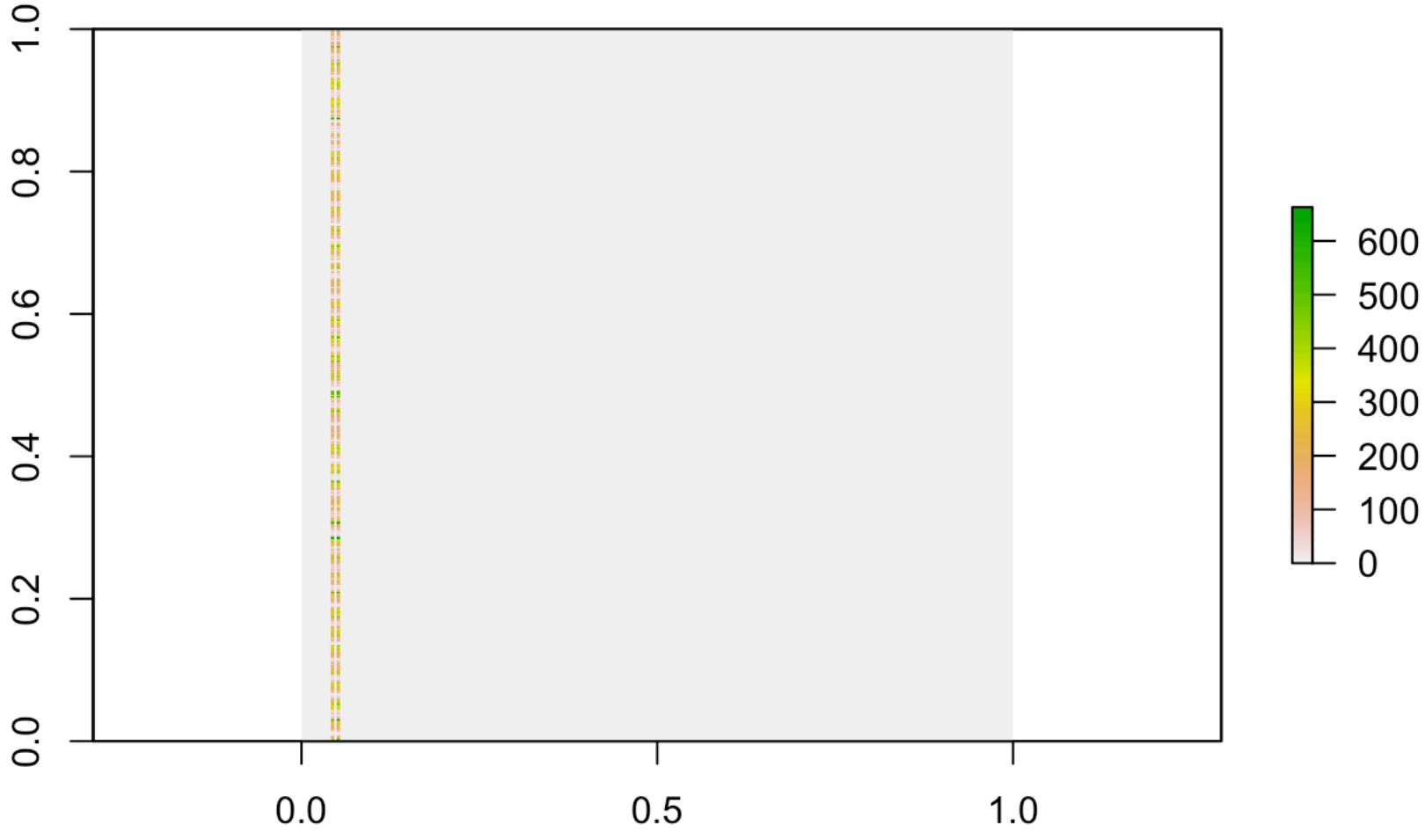
First, we are going to define our study area. To do this we are going to manipulate the USA state shapefile that we downloaded previously. We first need to transform our spatial coordinate system to match the raster data. Doing this tranformation on the vector file is much faster and more accurate than trying to reproject our raster time series. We defined the spatial projection earlier as WGS 84 (lat/long). We then want to make sure we just have the lower 48 states. For our final analysis we are going to look at regional variations in the data so we define those regions here. Because the `sf` object plays nicely with the `tidyverse` all of these dataframe manipulations are as easy on a spatial dataframe as they would be on a non-spatial dataframe. This is a huge improvement and difference compared to using the `sp` package for vector shapefiles. Using the `dply::group_by` function we can dissolve our polygons based on an attribute. In this case we can dissolve by our defined regions to remove state boundaries.

```
# Define our spatial projection
proj <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"
# Opening and cleaning the states layer
study_area <- sf::st_read(file.path(us_dir, "cb_2016_us_state_20m.shp"), quiet=TRUE)
%>%
  st_transform(proj) %>% #transform to match raster projection
  rename_all(tolower) %>% # rename all fields to lowercase for ease of coding later
  dplyr::filter(stusps %in% c("CO")) %>%
  mutate(lon = map_dbl(geometry, ~st_centroid(.x)[[1]]),
         lat = map_dbl(geometry, ~st_centroid(.x)[[2]]))

ecoregions <- sf::st_read(file.path(ecoregion_dir, "NA_CEC_Eco_Level1.shp"), quiet=TR
UE) %>%
  st_transform(st_crs(study_area)) %>%  # e.g. US National Atlas Equal Area
  dplyr::select(NA_L1NAME) %>%
  rename_all(tolower) %>% # rename all fields to lowercase for ease of coding later
  st_intersection(., st_union(study_area)) %>%
  group_by(na_l1name) %>%
  summarise %>%
  st_cast('MULTIPOLYGON')

ggplot() +
  geom_sf(data = ecoregions, aes(fill = na_l1name)) +
  geom_sf(data = study_area, color = 'black', alpha = 0, size = 1) +
  geom_text(data = study_area, aes(label = name, x = lon, y = lat), size = 7) +
  theme_minimal() +
  theme_map()
```



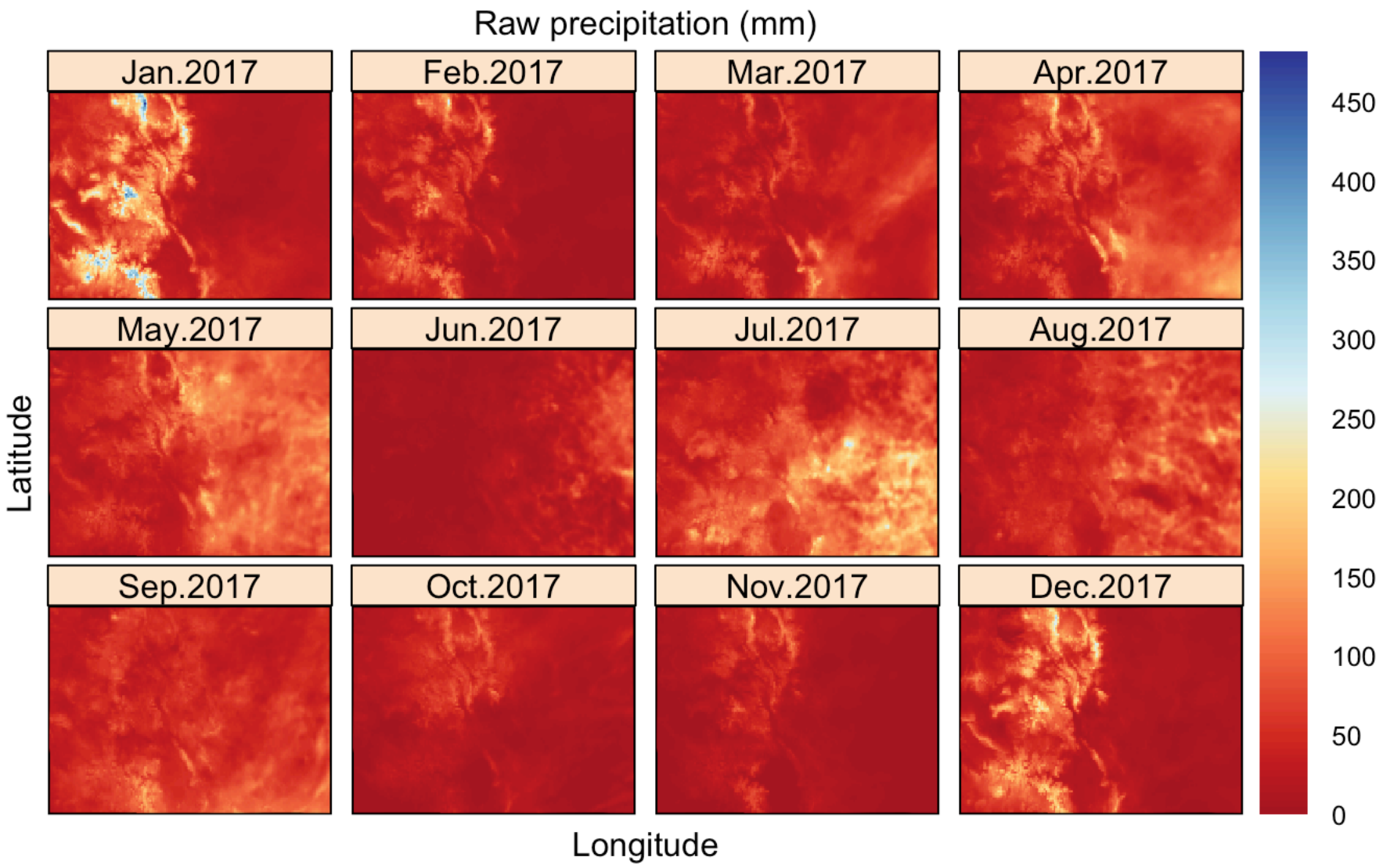
Now that we have our study area defined and prepped, we are going to import the precipitation data. These data are in NetCDF format, which is simply a multidimensional data file. Contained within this one NetCDF file is the attribute (x), space (y), and time (z) components of this time series. They are highly compressible and easily distributed. This particular NetCDF cannot be read in through the `raster` package, but the majority of NetCDFs can be read in throug the `raster` package, so we are going to use the `ncdf4` package to accomplish this import task.
Upon importing, NetCDFs are notorius for inverted dimensions, which can make the import seem like it has failed. For instance, if we were to import the precipitation data and plot the first time step (Jan 1979) this is the resulting raster would look like…

```
# Open the NetCDF file
nc <- nc_open(file.path(climate_dir, 'pr_gridMET.nc'))
# Unpack and transform the NetCDF file
inverted_netcdf <- attributes(nc$var)$names %>%
  ncvar_get(nc, .) %>%
  raster::brick(., crs= proj) # mask out to just the conterminous US
plot(inverted_netcdf[[1]])
```



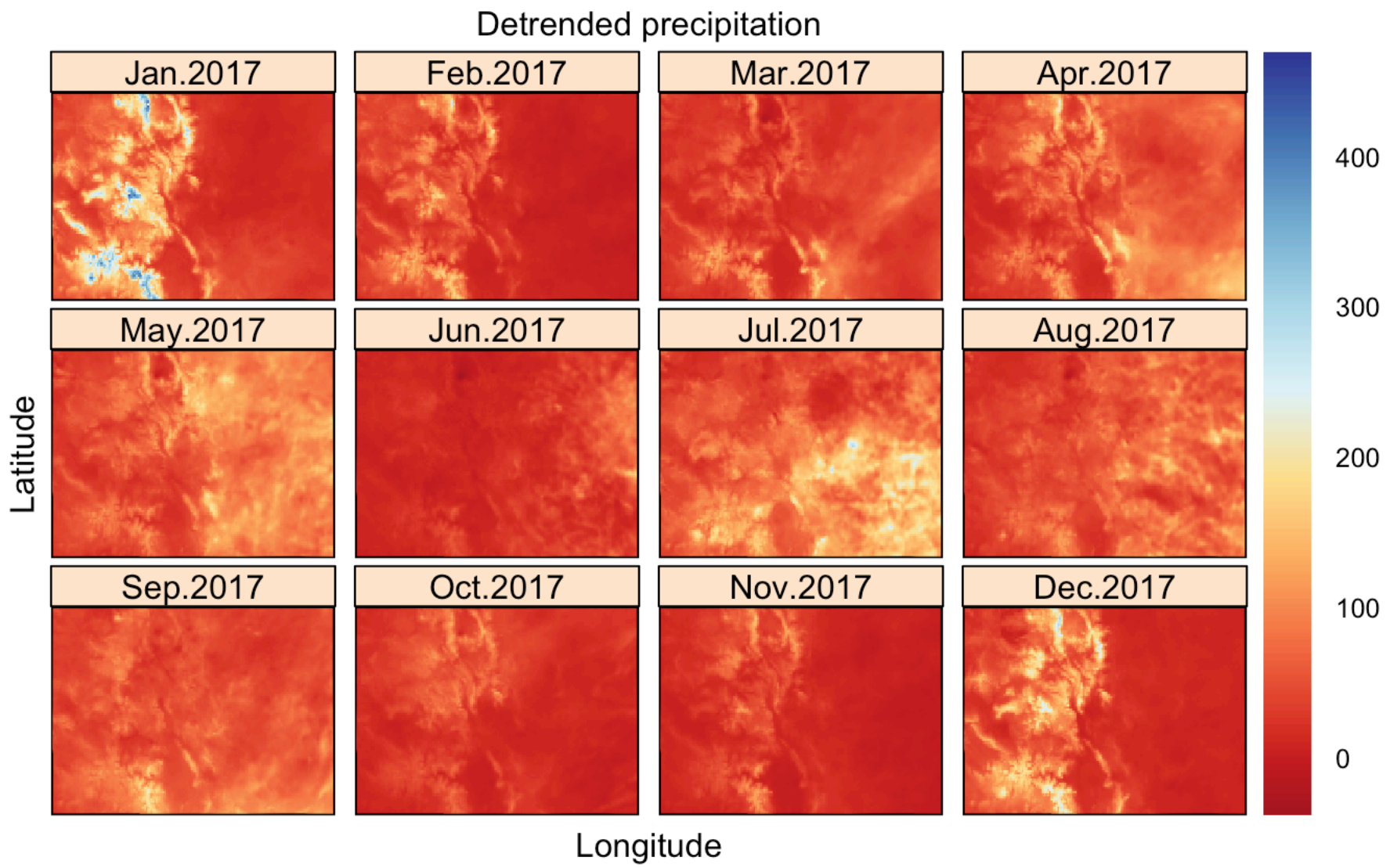
Clearly the dimensions are not displaying in the proper orientation. To correct that we are going to transpose the 3 dimensions (attribute, space, time) to reflect the correct order. One quick note/tip. You may have noticed that I am using the `%>%` operator. This is called a pipe and allows outputs of one task to be piped directly into the next operation without creating multiple intermediate objects. By piping the code becomes much more readable and clean. By importing the `tidyverse` you have automatically loaded the package, but within the `tidyverse` contains the `magrittr` package where the piping operator lives. You can use the piping operator outside of tidyverse operations, for instance below you will notice that we used this to pipe through the NetCDF import and into the `raster::brick` operator. Quite handly!

```
precip_series <-  attributes(nc$var)$names %>%
  ncvar_get(nc, .) %>%
  aperm(., c(2,3,1)) %>% # Transpose the dimensions in the correct order.  This takes some trial and error
  raster::brick(., crs= proj) # Create a raster time series stack that we can use for analysis

extent(precip_series) <- c(-124.793, -67.043, 25.04186, 49.41686) #set spatial extents
precip_series <- precip_series %>%
  crop(as(study_area, 'Spatial')) %>%
  mask(as(study_area, 'Spatial')) # mask out to just the Colorado and New Mexico US

# Set the names of the raster stack
precip_series <- set_names(precip_series, '1979-01-01', '2017-12-01')

raster_plot(input_series = precip_series[[457:468]], layout_vals = c(4,3),  title = 'Raw precipitation (mm)')
```



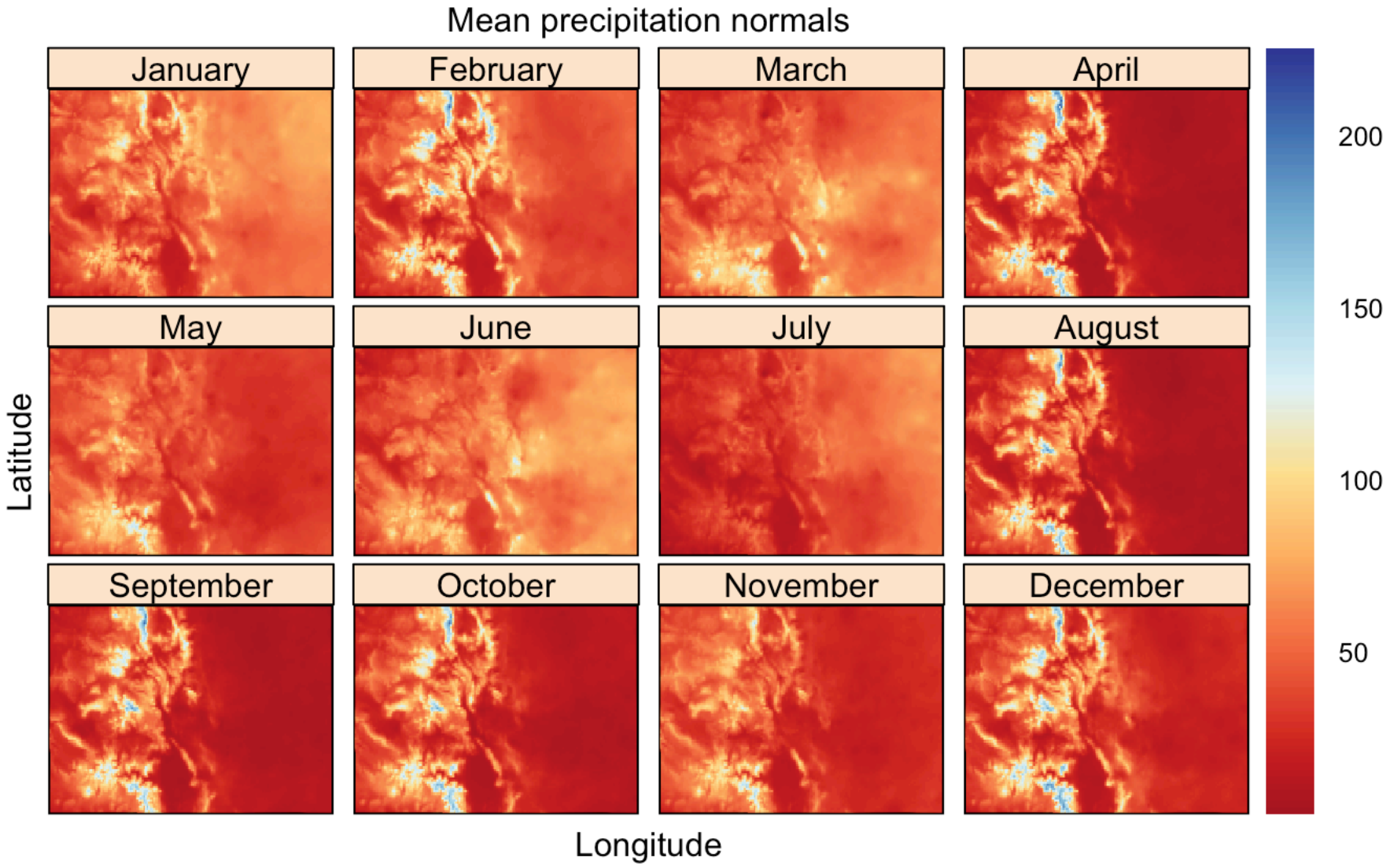
We know that there is inherent seasonality within our precipitation data. One way to remove the seasonality is to detrend the time series. To accomplish this we can simply extract the residuals from a linear regression between the series (the dependent variable) and time (the independent variable).

```
time <- 1:nlayers(precip_series) # Create our independent variable
detrended_precip <- calc(precip_series, get_residuals) # Create our residual (detrend
ed) time series stack
detrended_precip <- set_names(detrended_precip, '1979-01-01', '2017-12-01')

raster_plot(input_series = detrended_precip[[457:468]], layout_vals = c(4,3), title =
'Detrended precipitation')
```

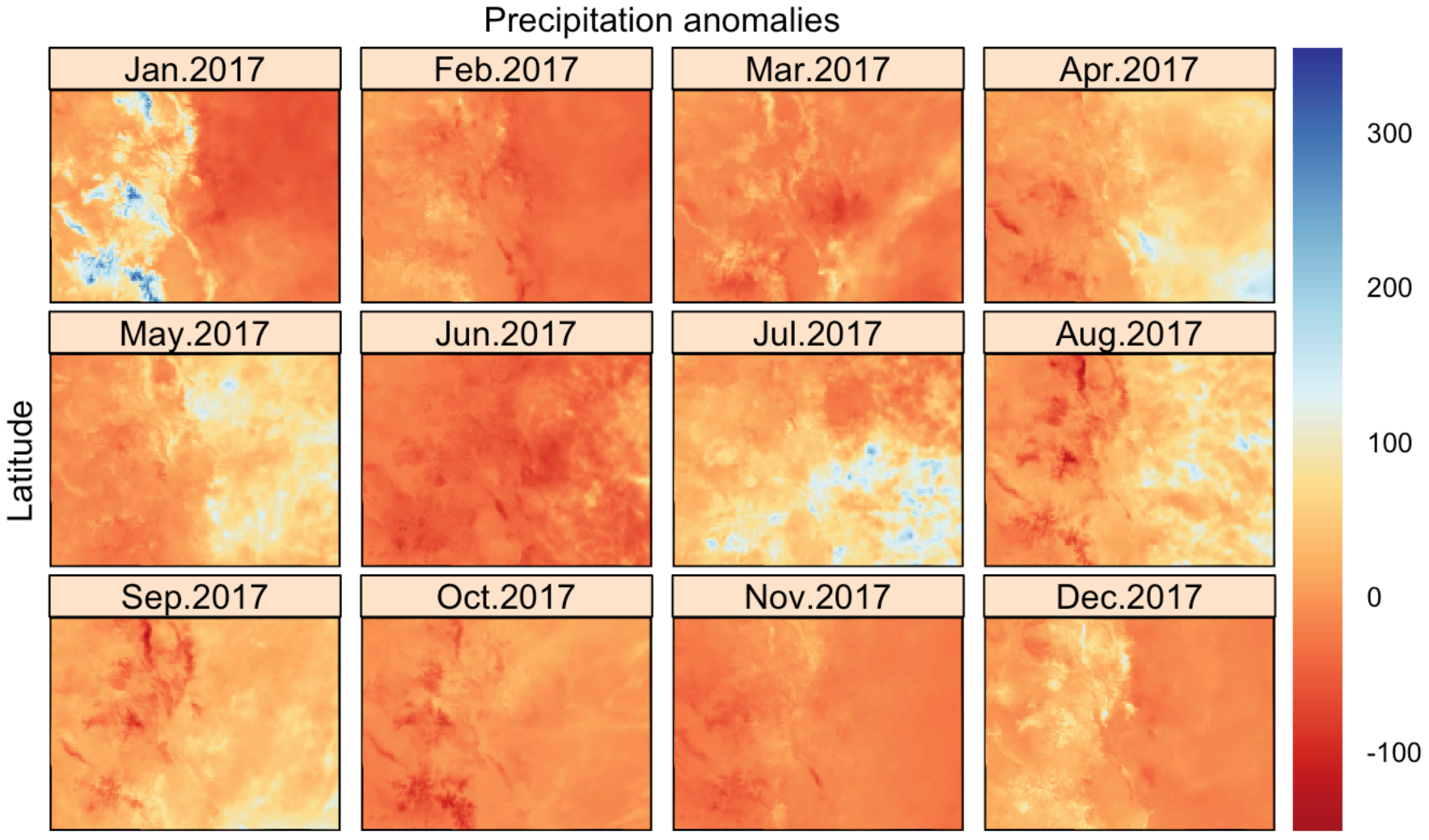


Ok, so now that we have created our detrened precipitation time series, let's create our precipitation anomalies using those detrended data.

```
month_names <- as.vector(unique(month(get_months('1979-01-01', '2017-12-01'), label =
TRUE, abbr = FALSE)))

# Creating our mean climatologies (long-term means) from the detrened series
mean_climatology <- zApply(detrended_precip, by = months, mean, name = month.abb[])
mean_climatology <- subset(mean_climatology, month_names) # Reorder the climatology f
rom alaphabetical to monthly
raster_plot(input_series = mean_climatology, layout_vals = c(4,3), title = 'Mean prec
ipitation normals')
```

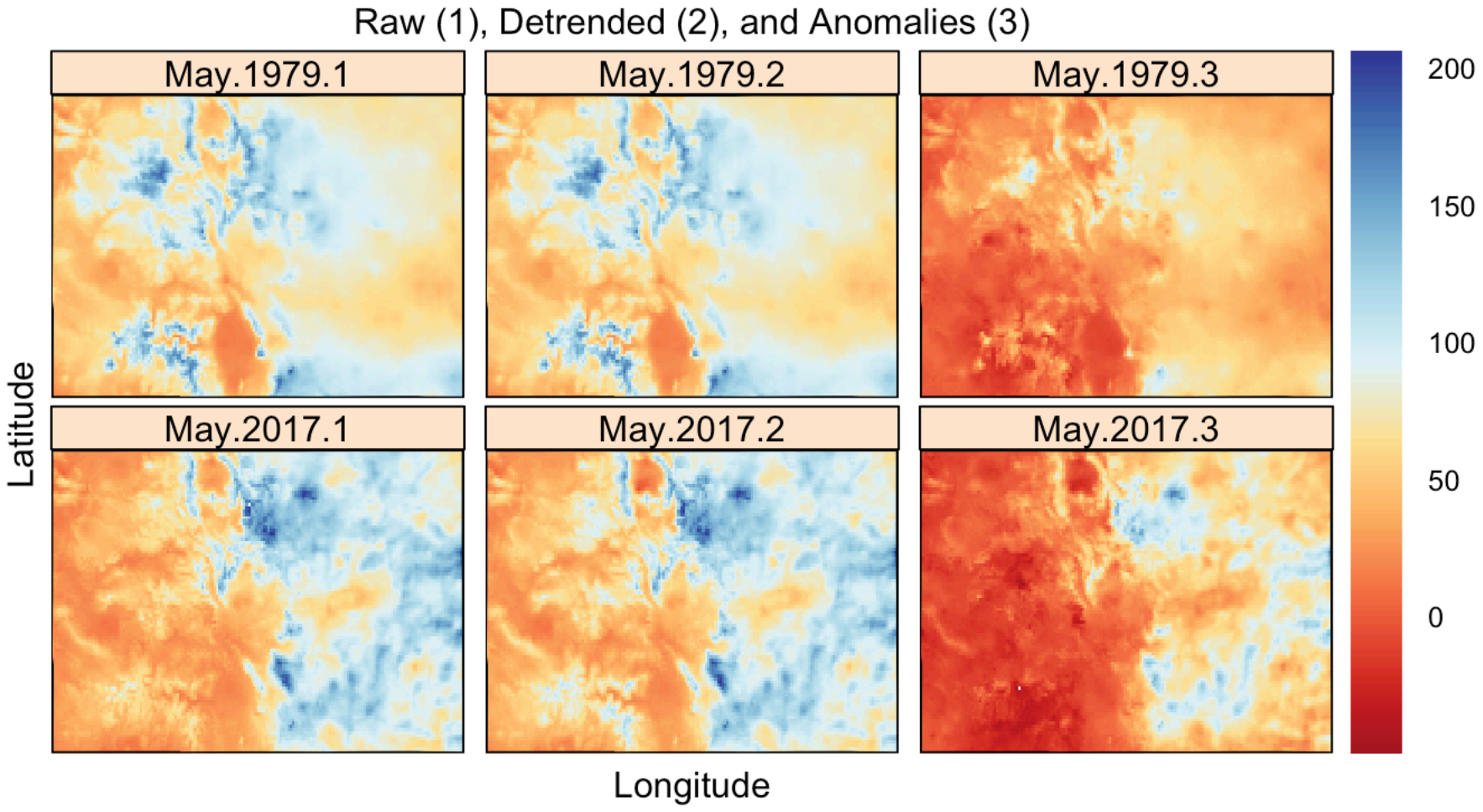



```
# Create precipitation anomalies
precip_anom <- overlay(x = detrended_precip, y = mean_climatology, fun = anom_fun)
precip_anom <- set_names(precip_anom, '1979-01-01', '2017-12-01')
raster_plot(input_series = precip_anom[[457:468]], layout_vals = c(4,3), title = 'Pre
cipitation anomalies')
```

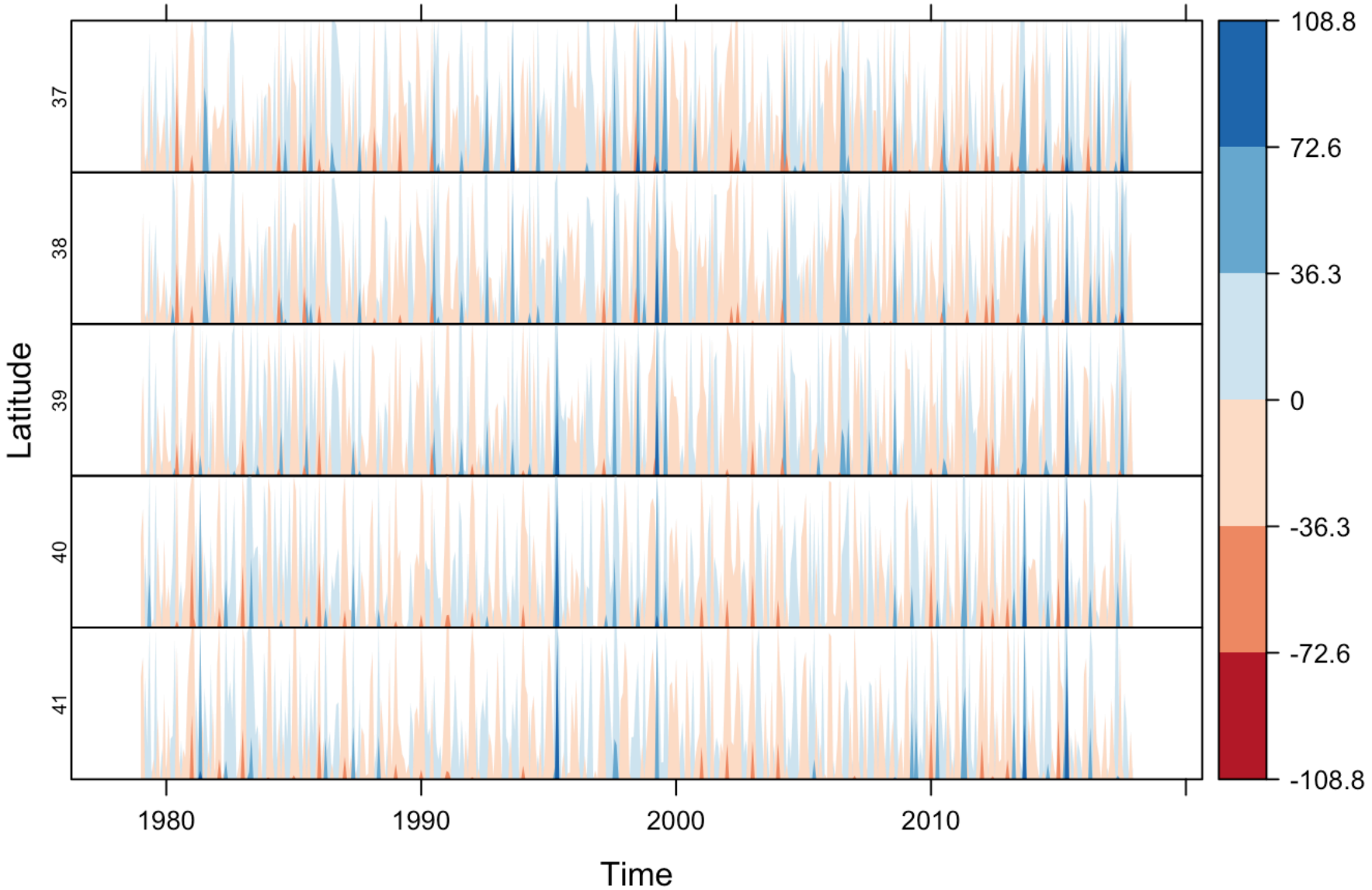


Now that all of the data are processed, how do those time series look spatially and temporally?

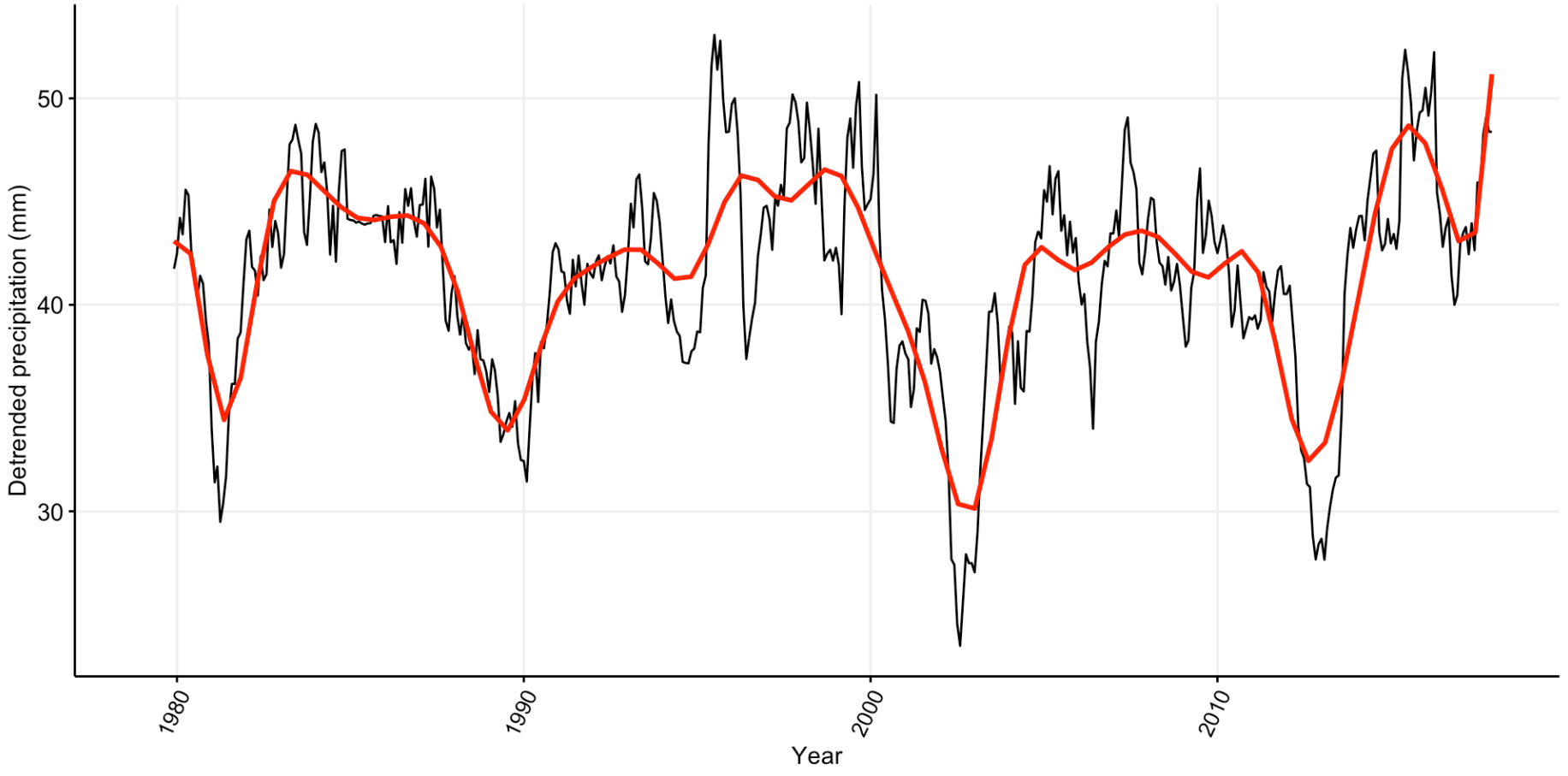
```
may_stack <- stack(precip_series[[5]], detrended_precip[[5]], precip_anom[[5]], preci
p_series[[461]], detrended_precip[[461]], precip_anom[[461]])

raster_plot(input_series = may_stack, layout_vals = c(3,2), title = 'Raw (1), Detrend
ed (2), and Anomalies (3)')
```



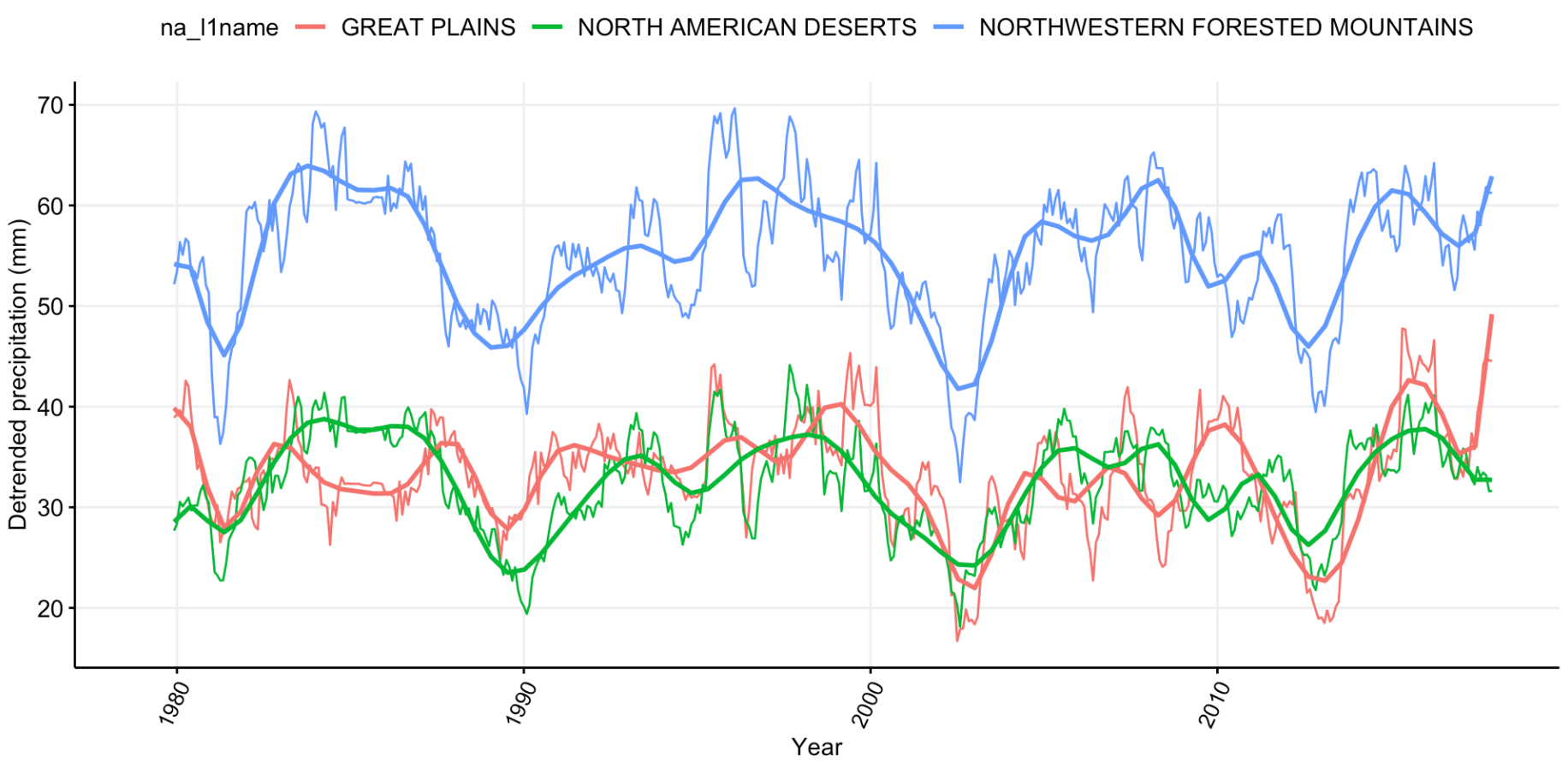

```
horizonplot(precip_anom, ylab="Latitude", xlab="Time",col.regions=(brewer.pal(n=6, 'RdBu')) )
```


Latitude
37
38
39
40
41
1980
1990
2000
2010
Time
108.8
72.6
36.3
0
-36.3
-72.6
-108.8

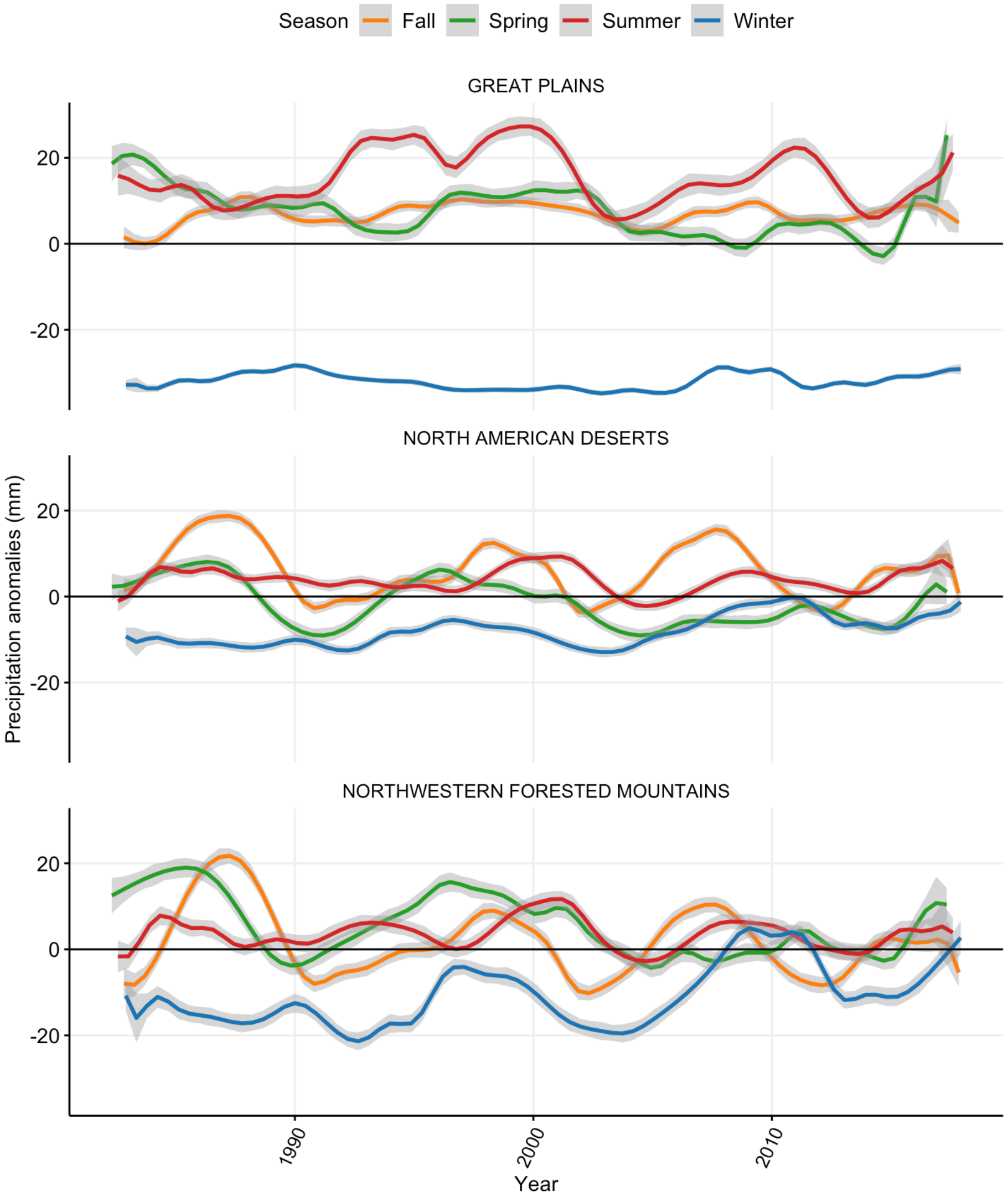
```
#Extract the mean precipitation per state
precip_co <- velox(detrended_precip)$extract(sp = study_area, fun = function(x) mean(
x, na.rm = TRUE), df = TRUE) %>%
    as_tibble()
# Rename to the raster layer names
colnames(precip_co) <- c('ID_sp', names(detrended_precip))
# Clean up the dataframe and create a 2-year rolling average
precip_co %>%
  mutate(stusps = as.data.frame(study_area)$stusps) %>%
  gather(key = key, value = precip_mean, -ID_sp, -stusps) %>%
  separate(key,
           into = c("months", 'year'),
           sep = "\\.") %>%
  mutate(year_mon = as.Date(as.yearmon(paste0(months, ' ', year)))) %>%
  dplyr::select(year_mon, precip_mean) %>%
  arrange(year_mon) %>%
  mutate(yearly_mean = rollmean(precip_mean, 12, align='right', fill = NA )) %>%
  ggplot(aes(x = year_mon, y = yearly_mean)) +
  geom_line() +
  geom_smooth(method = lm, formula = y ~ splines::bs(x, 30), se = FALSE, color = 'red
') +
  xlab("Year") + ylab("Detrended precipitation (mm)") +
  theme_pub()
```



```
#Extract the mean precipitation per state
precip_eco <- velox(detrended_precip)$extract(sp = ecoregions, fun = function(x) mean
(x, na.rm = TRUE), df = TRUE) %>%
    as_tibble()
# Rename to the raster layer names
colnames(precip_eco) <- c('ID_sp', names(detrended_precip))

# Clean up the dataframe and create a 2-year rolling average
precip_eco %>%
  mutate(na_l1name = as.data.frame(ecoregions)$na_l1name) %>%
  gather(key = key, value = precip_mean, -ID_sp, -na_l1name) %>%
  separate(key,
           into = c("months", 'year'),
           sep = "\\.") %>%
  mutate(year_mon = as.Date(as.yearmon(paste0(months, ' ', year)))) %>%
  dplyr::select(na_l1name, year_mon, precip_mean) %>%
  arrange(na_l1name, year_mon) %>%
  group_by(na_l1name) %>%
  mutate(yearly_mean = rollmean(precip_mean, 12, align='right', fill = NA )) %>%
  ggplot(aes(x = year_mon, y = yearly_mean, group = na_l1name, color = na_l1name)) +
  geom_line() +
  geom_smooth(method = lm, formula = y ~ splines::bs(x, 30), se = FALSE) +
  xlab("Year") + ylab("Detrended precipitation (mm)") +
  theme_pub() +
  theme(legend.position = 'top',
        legend.direction = "horizontal")
```



```
#Extract the mean precipitation per state
precip_ecoregions <- velox(precip_anom)$extract(sp = ecoregions, fun = function(x) me
an(x, na.rm = TRUE), df = TRUE) %>%
    as_tibble()
# Rename to the raster layer names
colnames(precip_ecoregions) <- c('ID_sp', names(precip_anom))
# Clean up the dataframe and create a 2-year rolling average
precip_ecoregions %>%
  mutate(na_l1name = as.data.frame(ecoregions)$na_l1name) %>%
  gather(key = key, value = precip_mean, -ID_sp, -na_l1name) %>%
  separate(key,
           into = c("months", 'year'),
           sep = "\\.") %>%
  mutate(year_mon = as.Date(as.yearmon(paste0(months, ' ', year))),
         seasons = as.factor(classify_seasons(months))) %>%
  dplyr::select(na_l1name, seasons, year_mon, precip_mean) %>%
  arrange(na_l1name, seasons, year_mon) %>%
  group_by(na_l1name, seasons) %>%
  mutate(yearly_mean = rollmean(precip_mean, 12, align='right', fill = NA )) %>%
  ggplot(aes(x = year_mon, y = yearly_mean, group = factor(seasons), color = factor(s
easons))) +
  geom_smooth(method = lm, formula = y ~ splines::bs(x, 30)) +
  geom_hline(yintercept = 0, color = 'black')  +
  xlab("Year") + ylab("Precipitation anomalies (mm)") +
  scale_color_manual(values = c("Winter" = "#1F77B4",
                                "Spring" = "#2CA02C",
                                "Summer" =  "#D62728",
                                "Fall" = "#FF7F0E"),
                     name="Season") +
theme_pub() +
theme(legend.position = 'top',
      legend.direction = "horizontal") +
facet_wrap(~na_l1name, ncol = 1)
```


Season
Fall
Spring
Summer
Winter
GREAT PLAINS
NORTH AMERICAN DESERTS
NORTHWESTERN FORESTED MOUNTAINS
Precipitation anomalies (mm)
20
0
-20
1990
2000
2010
Year