Project 1

Precipitation Downscaling

Coleman Nickum (CN33)

2023-11-13

Table of contents

[1. Setup 1](#_Toc150733233)

[2. Precipitation Data: ERA5 Reanalysis Dataset 2](#_Toc150733234)

[2.1 Precipitation Heatmap: Past vs Present 3](#_Toc150733235)

[3. Temperature Data: ERA5 Reanalysis Dataset 4](#_Toc150733236)

[3.1 Temperature Heatmap: Past vs Present 5](#_Toc150733237)

[3.2 Ensuring temperature and precipitation correspond to the same time period 6](#_Toc150733238)

[4. Splitting Data: Training Period vs. Testing Period 6](#_Toc150733239)

[4.1 Testing and Training Period Variables 6](#_Toc150733240)

[5. Preprocessing Data: Climatology and Mean Centering 7](#_Toc150733241)

[5.1 Apply to Training and Testing Datasets 7](#_Toc150733242)

[6. Principle Component Analysis 7](#_Toc150733243)

[6.1 Fitting 7](#_Toc150733244)

[6.2 Plotting Variance Explained by PCs 7](#_Toc150733245)

[6.3 Plotting Spatial Patterns 8](#_Toc150733246)

[6.4 Plotting Time Series 8](#_Toc150733247)

[7. KNN: Resampling Algorithm 9](#_Toc150733248)

[7.1 Combining KNN and PCA 9](#_Toc150733249)

[8.2 Predicted vs. Actual Values 10](#_Toc150733250)

[9. Quantile-Quantile Mapping: Bias Correction 11](#_Toc150733251)

# 1. Setup

using Dates  
using MultivariateStats  
using Plots  
using NCDatasets  
using StatsBase  
using Unitful  
using DataFrames  
using Statistics  
Plots.default(; margin=4Plots.mm, size=(700, 400), linewidth=2)

# 2. Precipitation Data: ERA5 Reanalysis Dataset

# Dataset covers daily precipitation (mm) from 1/1/1979 to 12/31/2020  
ds\_precip = NCDataset("/Users/coleman/Documents/GitHub/project-01-precipitation-downscaling-ColemanNickum/data/raw/precip.nc")  
println(ds\_precip)

Dataset: /Users/coleman/Documents/GitHub/project-01-precipitation-downscaling-ColemanNickum/data/raw/precip.nc  
Group: /  
  
Dimensions  
 lon = 10  
 lat = 10  
 time = 15341  
  
Variables  
 lon (10)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lon  
 Attributes:  
 \_FillValue = NaN  
 long\_name = Longitude  
 units = degrees\_east  
 axis = X  
 standard\_name = longitude  
 actual\_range = Float32[0.25, 359.75]  
 coordinate\_defines = center  
  
 lat (10)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lat  
 Attributes:  
 \_FillValue = NaN  
 actual\_range = Float32[89.75, -89.75]  
 long\_name = Latitude  
 units = degrees\_north  
 axis = Y  
 standard\_name = latitude  
 coordinate\_defines = center  
  
 time (15341)  
 Datatype: Union{Missing, DateTime} (Float64)  
 Dimensions: time  
 Attributes:  
 \_FillValue = NaN  
 long\_name = Time  
 axis = T  
 standard\_name = time  
 coordinate\_defines = start  
 actual\_range = [692496.0, 701232.0]  
 delta\_t = 0000-00-01 00:00:00  
 avg\_period = 0000-00-01 00:00:00  
 \_ChunkSizes = 1  
 units = hours since 1900-01-01  
 calendar = proleptic\_gregorian  
  
 precip (10 × 10 × 15341)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lon × lat × time  
 Attributes:  
 \_FillValue = -9.96921e36  
 var\_desc = Precipitation  
 level\_desc = Surface  
 statistic = Total  
 parent\_stat = Other  
 long\_name = Daily total of precipitation  
 cell\_methods = time: sum  
 avg\_period = 0000-00-01 00:00:00  
 actual\_range = Float32[0.0, 428.02423]  
 units = mm  
 valid\_range = Float32[0.0, 1000.0]  
 dataset = CPC Global Precipitation  
 \_ChunkSizes = Int32[1, 360, 720]  
 missing\_value = -9.96921e36

precip\_time = ds\_precip["time"][:];  
precip\_lon = ds\_precip["lon"][:];  
precip\_lat = ds\_precip["lat"][:];  
precip = ds\_precip["precip"][:,:,:];

precip = precip .\* 1u"mm";

precip\_lat = reverse(precip\_lat)  
precip = reverse(precip, dims=2);

## 2.1 Precipitation Heatmap: Past vs Present

h1 = heatmap(  
 precip\_lon,  
 precip\_lat,  
 precip[:, :, 1]';  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Precipitation on 1/1/1979"  
)  
  
h2 = heatmap(  
 precip\_lon,  
 precip\_lat,  
 precip[:, :, end]';  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Precipitation on 12/31/2020"  
)  
plot(h1, h2; layout=(1, 2), size=(900, 400))

close(ds\_precip)

closed Dataset

# 3. Temperature Data: ERA5 Reanalysis Dataset

# Dataset covers daily temperature (Kelvin) from 1/1/1979 to 12/31/2020  
ds\_temp = NCDataset("/Users/coleman/Documents/GitHub/project-01-precipitation-downscaling-ColemanNickum/data/raw/temp.nc")  
println(ds\_temp)

Dataset: /Users/coleman/Documents/GitHub/project-01-precipitation-downscaling-ColemanNickum/data/raw/temp.nc  
Group: /  
  
Dimensions  
 longitude = 21  
 latitude = 33  
 time = 15341  
  
Variables  
 longitude (21)  
 Datatype: Union{Missing, Float64} (Float64)  
 Dimensions: longitude  
 Attributes:  
 \_FillValue = NaN  
  
 latitude (33)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: latitude  
 Attributes:  
 \_FillValue = NaN  
 units = degrees\_north  
 long\_name = latitude  
  
 time (15341)  
 Datatype: DateTime (Int64)  
 Dimensions: time  
 Attributes:  
 units = days since 1979-01-01 00:00:00  
 calendar = proleptic\_gregorian  
  
 t2m (21 × 33 × 15341)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: longitude × latitude × time  
 Attributes:  
 \_FillValue = NaN  
 units = K  
 long\_name = 2 metre temperature

temp\_time = ds\_temp["time"][:];  
temp\_lon = ds\_temp["longitude"][:];  
temp\_lat = ds\_temp["latitude"][:];  
temp = ds\_temp["t2m"][:,:,:];

temp\_lat = reverse(temp\_lat)  
temp = reverse(temp, dims=2);

## 3.1 Temperature Heatmap: Past vs Present

h1 = heatmap(  
 temp\_lon,  
 temp\_lat,  
 temp[:, :, 1]';  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Temperature on 1/1/1979"  
)  
  
h2 = heatmap(  
 temp\_lon,  
 temp\_lat,  
 temp[:, :, end]';  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Temperature on 12/31/2020"  
)  
plot(h1, h2; layout=(1, 2), size=(900, 400))

## 3.2 Ensuring temperature and precipitation correspond to the same time period

@assert temp\_time == precip\_time

time\_data = Dates.Date.(temp\_time)

15341-element Vector{Date}:  
 1979-01-01  
 1979-01-02  
 1979-01-03

# 4. Splitting Data: Training Period vs. Testing Period

# Performing a typical split ratio (roughly 80:20) for training and testing data collected from 1979 to 2021. With 42 years of data, the training dataset will encompass 32 years and the testing period will be 10 years (2009 to 2021)  
split\_idx = findfirst(time\_data .== time\_data[end] - Dates.Year(10))  
train\_idx = 1:split\_idx  
test\_idx = (split\_idx+1):length(time\_data);

## 4.1 Testing and Training Period Variables

precip\_train = precip[:, :, train\_idx];  
precip\_test = precip[:, :, test\_idx];  
temp\_train = temp[:, :, train\_idx];  
temp\_test = temp[:, :, test\_idx];

# 5. Preprocessing Data: Climatology and Mean Centering

function preprocess(temp::Array{T, 3}, temp\_reference::Array{T, 3})::AbstractMatrix where T  
#Computing anomalies that removes climatology from our matrix to allow greater clarity of temperature variations by removing the influence of climate patterns  
climatology = mean(temp\_reference, dims=3)  
anomalies = temp .- climatology  
  
#Reshaping our temperature dataset to produce a 2D matrix of time (rows) and locations (columns):   
temp\_mat = reshape(anomalies, size(temp, 1) \* size(temp, 2), size(temp, 3))  
 return temp\_mat  
end

preprocess (generic function with 1 method)

## 5.1 Apply to Training and Testing Datasets

#Preprocessing temp\_train and temp\_test; both are preprocessed to the temp training data so they both use the same climatology  
n\_lon, n\_lat, n\_t = size(temp)  
temp\_training\_matrix = preprocess(temp\_train, temp\_train);  
temp\_testing\_matrix = preprocess(temp\_test, temp\_train);

# 6. Principle Component Analysis

## 6.1 Fitting

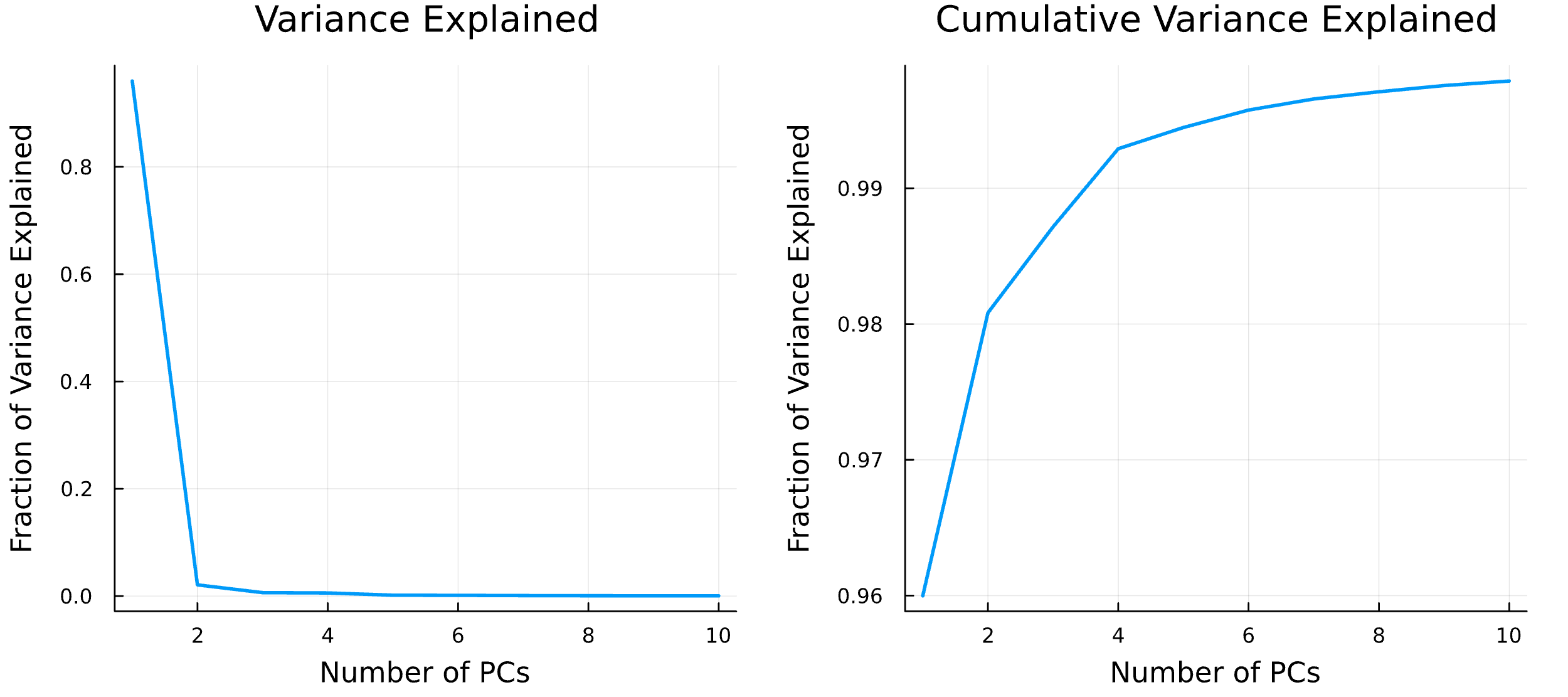
#Fitting a PCA model to training period to automatically choose the number of principle components  
PCA\_model = fit(PCA, temp\_training\_matrix; maxoutdim=10, pratio=0.999);

## 6.2 Plotting Variance Explained by PCs

# Variables for plotting the variance accounted by the principle components   
variance\_explained = principalvars(PCA\_model)  
total\_var = var(PCA\_model)  
cumulative\_var = cumsum(variance\_explained)./total\_var

10-element Vector{Float32}:

p1 = plot(  
 variance\_explained / total\_var;  
 xlabel="Number of PCs",  
 ylabel="Fraction of Variance Explained",  
 label=false,  
 title="Variance Explained"  
)  
p2 = plot(  
 cumulative\_var;  
 xlabel="Number of PCs",  
 ylabel="Fraction of Variance Explained",  
 label=false,  
 title="Cumulative Variance Explained"  
)  
plot(p1, p2; layout=(1, 2), size=(900, 400))



## 6.3 Plotting Spatial Patterns

p = []  
for i in 1:4  
 pc = projection(PCA\_model)[:, i]  
 pc\_reshaped = reshape(pc, n\_lon, n\_lat)'  
 pi = heatmap(  
 temp\_lon,  
 temp\_lat,  
 pc\_reshaped;  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="PC $i",  
 aspect\_ratio=:equal,  
 cmap=:PuOr  
 )  
 push!(p, pi)  
end  
plot(p...; layout=(1, 4), size=(1500, 600))

## 6.4 Plotting Time Series

pc\_ts = predict(PCA\_model, temp\_training\_matrix)  
Months = Dates.month.(time\_data)  
custom\_xticks = [1, 3, 6, 9, 12]  
p = []  
for i in 1:4  
 pi = scatter(  
 Months,  
 pc\_ts[i, :];  
 xlabel="Months in Year",  
 ylabel="PC $i",  
 title="PC $i",  
 label=false,  
 alpha=0.3,  
 color=:blue,  
 xticks=(custom\_xticks, custom\_xticks)  
 )  
 push!(p, pi)  
end  
plot(p...; layout=(1, 4), size=(1100, 600))

# 7. KNN: Resampling Algorithm

function euclidean\_distance(x::AbstractVector, y::AbstractVector)::AbstractFloat  
 return sqrt(sum((x .- y) .^ 2))  
end  
  
function nsmallest(x::AbstractVector, n::Int)::Vector{Int}  
 idx = sortperm(x)  
 return idx[1:n]  
end  
  
function knn(X::AbstractMatrix, X\_i::AbstractVector, K::Int)::Tuple{Int,AbstractVector}  
 # calculate the distances between X\_i and each row of X  
 dist = [euclidean\_distance(X\_i, X[j, :]) for j in 1:size(X, 1)]  
 idx = nsmallest(dist, K)  
 w = 1 ./ dist[idx]  
 w ./= sum(w)  
 idx\_sample = sample(idx, Weights(w))  
 return (idx\_sample, vec(X[idx\_sample, :]))  
end

knn (generic function with 1 method)

## 7.1 Combining KNN and PCA

function predict\_knn(temp\_train, temp\_test, precip\_train; n\_pca::Int)  
 X\_train = preprocess(temp\_train, temp\_train)  
 X\_test = preprocess(temp\_test, temp\_train)  
  
 # fitting the PCA model to the training temperature data  
 pca\_model = fit(PCA, X\_train; maxoutdim=n\_pca)  
  
 # project the test data onto the PCA basis  
 train\_embedded = predict(pca\_model, X\_train)  
 test\_embedded = predict(pca\_model, X\_test)  
  
 #use the `knn` function for each point in the test data  
 precip\_pred = map(1:size(X\_test, 2)) do i  
 idx, \_ = knn(train\_embedded', test\_embedded[:, i], 3)  
 precip\_train[:, :, idx]  
 end  
  
 # return a matrix of predictions (precip\_pred)  
 return precip\_pred  
end

predict\_knn (generic function with 1 method)

## 7.2 Predicted vs. Actual Values

t\_sample = rand(1:size(temp\_test, 3), 4)  
precip\_pred = predict\_knn(temp\_train, temp\_test[:, :, t\_sample], precip\_train; n\_pca=4)  
  
p = map(eachindex(t\_sample)) do ti  
 t = t\_sample[ti]  
 y\_pred = precip\_pred[ti]'  
 y\_actual = precip\_test[:, :, t]'  
 cmax = max(maximum(skipmissing(y\_pred)), maximum(skipmissing(y\_actual)))  
 cmax = ustrip(u"mm", cmax)  
  
 p1 = heatmap(  
 precip\_lon,  
 precip\_lat,  
 y\_pred;  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Predicted",  
 aspect\_ratio=:equal,  
 clims=(0, cmax)  
 )  
 p2 = heatmap(  
 precip\_lon,  
 precip\_lat,  
 y\_actual;  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Actual",  
 aspect\_ratio=:equal,  
 clims=(0, cmax)  
 )  
 plot(p1, p2; layout=(2, 1), size=(1000, 400))  
end  
plot(p...; layout=(2, 3), size=(1400, 1200))

# 8. Quantile-Quantile Mapping: Bias Correction

function knn\_quantile\_mapping(temp\_train, temp\_test, precip\_train, precip\_test; n\_pca::Int, n\_trees::Int)  
 # KNN prediction  
 precip\_pred = predict\_knn(temp\_train, temp\_test, precip\_train, n\_pca=n\_pca)  
 y\_pred = precip\_pred[1] # Assumes the model returns y\_pred directly  
  
 # Actual precipitation values  
 y\_actual = precip\_test[:,:,1]  
  
 # Reshape y\_pred for quantile mapping  
 y\_pred\_corrected = reshape(y\_pred, length(precip\_lon)\*length(precip\_lat))  
  
 # Quantile mapping  
 q = 0.5   
 quantile\_pred = quantile(skipmissing(y\_pred), q)  
  
 for i in 1:length(y\_pred\_corrected)  
 if !ismissing(y\_pred\_corrected[i])  
 # Calculates the quantile of the predicted value in the predicted distribution  
 point\_quantile = searchsortedfirst(sort(y\_pred\_corrected), y\_pred\_corrected[i]) / length(y\_pred\_corrected)  
  
 # Identifies the quantile of the actual value in the observed distribution  
 quantile\_actual = quantile(skipmissing(y\_actual), point\_quantile)  
  
 # Adjust the predicted value using quantile mapping  
 y\_pred\_corrected[i] = quantile\_actual  
 end  
 end  
  
 # Reshape the corrected predictions  
 y\_pred\_corrected = reshape(y\_pred\_corrected, length(precip\_lon), length(precip\_lat))  
  
 return y\_pred\_corrected  
end

knn\_quantile\_mapping (generic function with 1 method)

t\_sample = rand(1:size(temp\_test, 3), 4)  
y\_pred\_corrected = knn\_quantile\_mapping(temp\_train, temp\_test[:, :, t\_sample], precip\_train, precip\_test; n\_pca=4, n\_trees=100)  
  
p = map(eachindex(t\_sample)) do ti  
 t = t\_sample[ti]  
 y\_actual = precip\_test[:, :, t]'  
 cmax = max(maximum(skipmissing(y\_pred\_corrected)), maximum(skipmissing(y\_actual)))  
 cmax = ustrip(u"mm", cmax)  
  
 p1 = heatmap(  
 precip\_lon,  
 precip\_lat,  
 y\_pred\_corrected';  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Predicted (Quantile Mapping)",  
 aspect\_ratio=:equal,  
 clims=(0, cmax)  
 )  
 p2 = heatmap(  
 precip\_lon,  
 precip\_lat,  
 y\_actual;  
 xlabel="Longitude",  
 ylabel="Latitude",  
 title="Actual",  
 aspect\_ratio=:equal,  
 clims=(0, cmax)  
 )  
 plot(p1, p2; layout=(2, 1), size=(1000, 400))  
end  
plot(p...; layout=(2, 3), size=(1400, 1200))