Land Cover Analysis

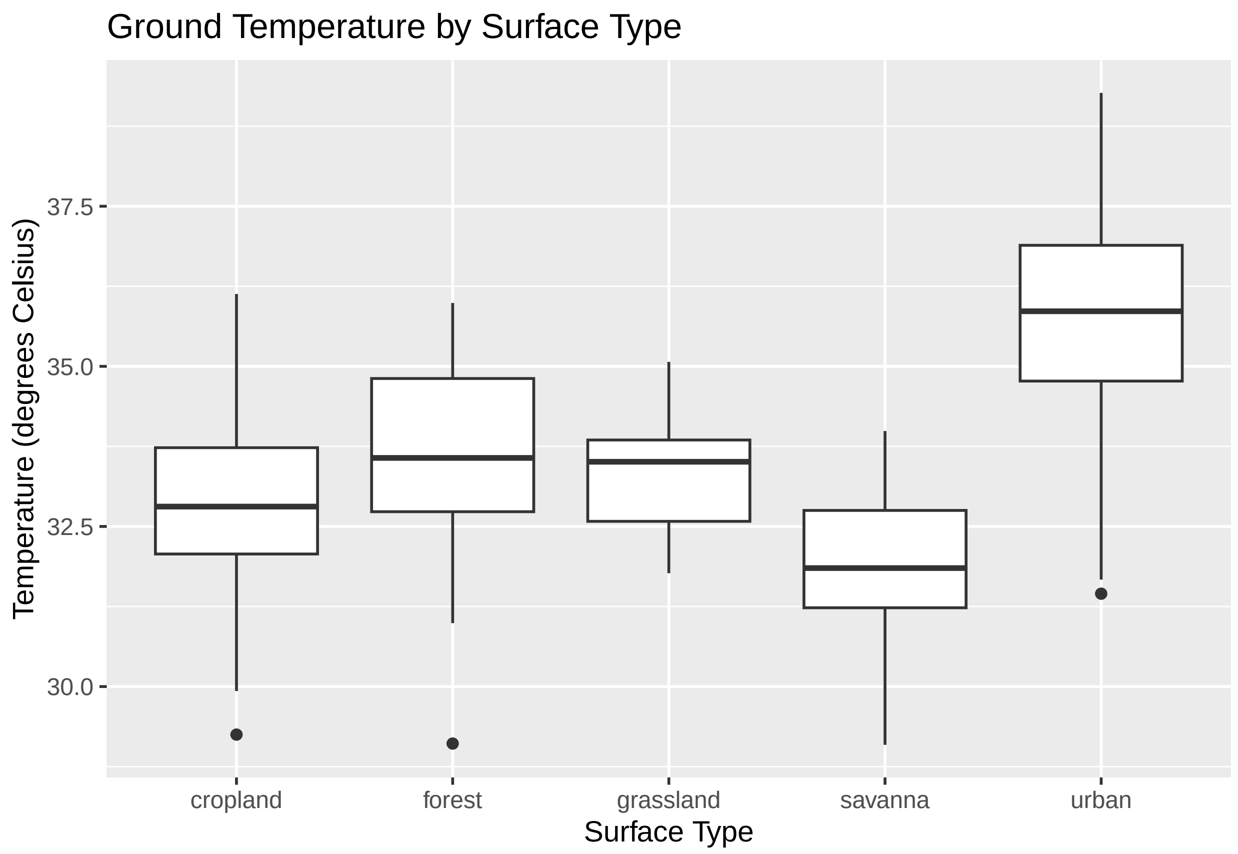
Jillian Warburton

2023-04-03

Exploratory Data Analysis

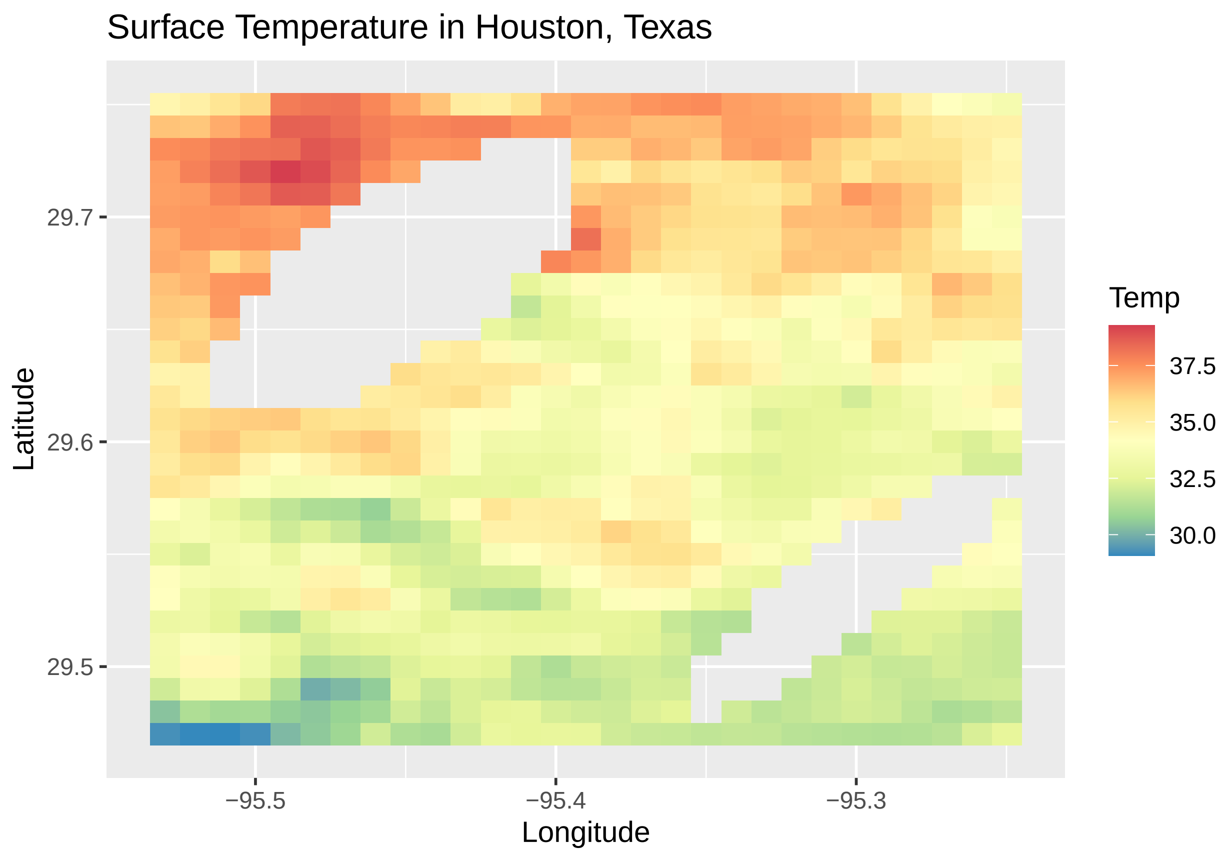
1. Create side-by-side boxplots of temperature by land cover type.

```r
ggplot(data = data, mapping = aes(x = Surface, y = Temp)) +
  geom_boxplot() +
  labs(title = "Ground Temperature by Surface Type",
       x = "Surface Type",
       y = "Temperature (degrees Celsius)")
```

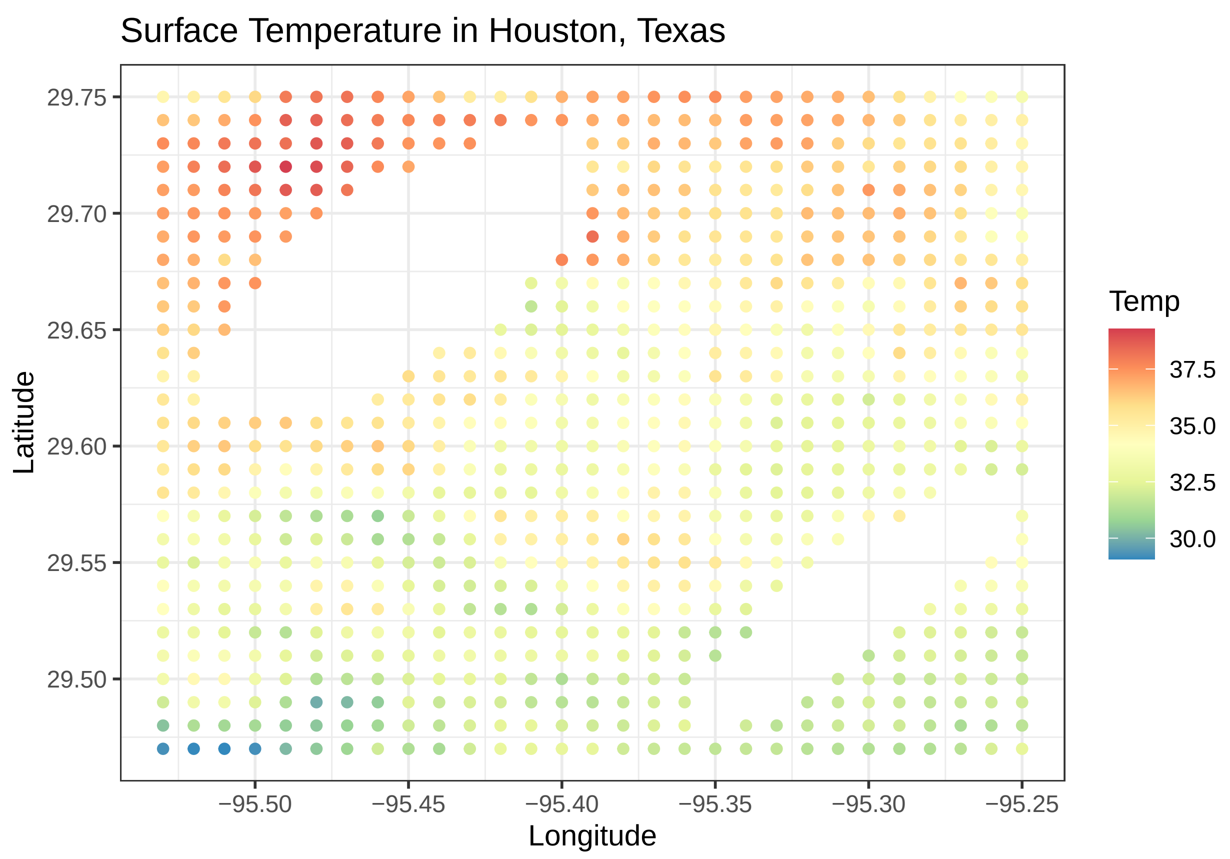


2. Create of heat map of temperature over Houston using `geom_raster()` AND `geom_point()`.

```r
ggplot(data = data, mapping = aes(x = Lon, y = Lat, fill = Temp)) +
  geom_raster() +
  scale_fill_distiller(palette = "Spectral", na.value = NA) +
  labs(title = "Surface Temperature in Houston, Texas",
       x = "Longitude",
       y = "Latitude")
```

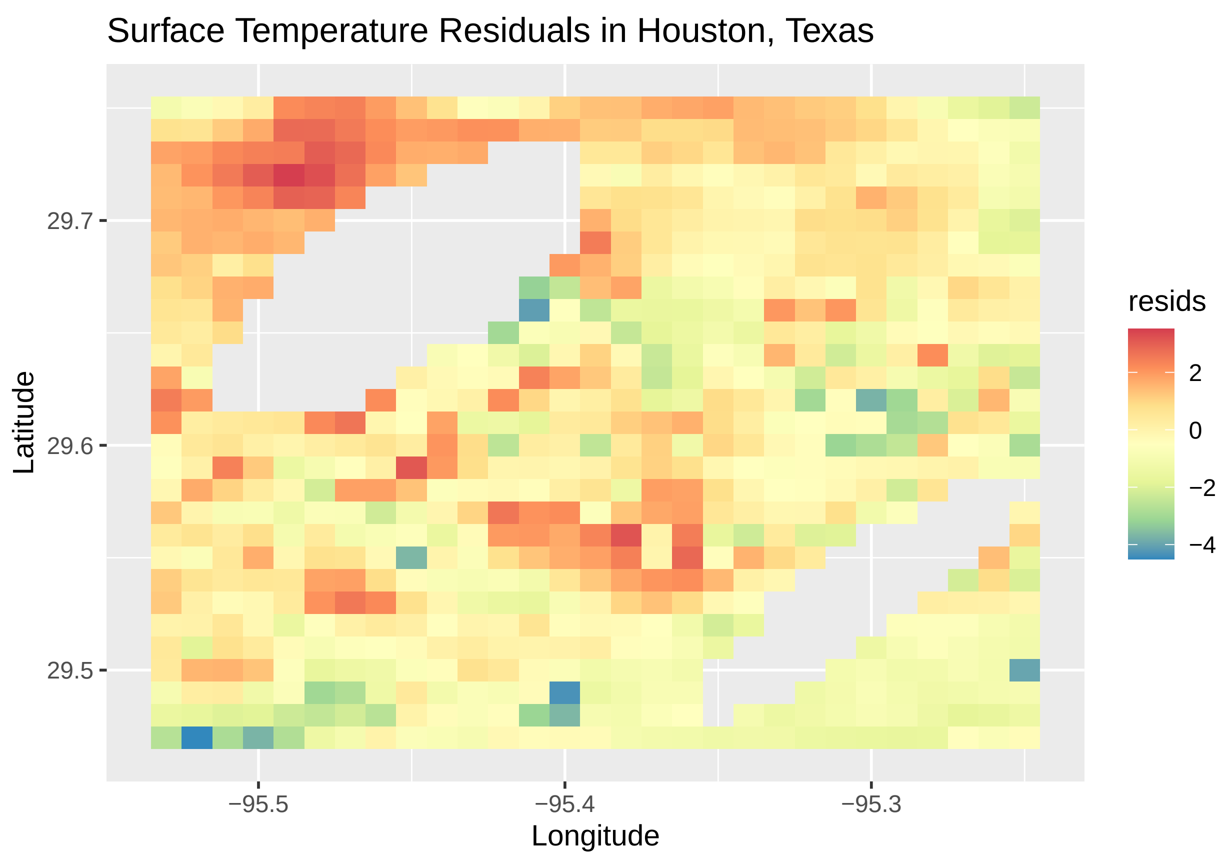


```r
ggplot(data = data , mapping = aes(x = Lon, y = Lat, color = Temp)) +
  geom_point() +
  scale_color_distiller(palette = "Spectral", na.value = NA) +
  theme_bw() +
  labs(title = "Surface Temperature in Houston, Texas",
       x = "Longitude",
       y = "Latitude")
```



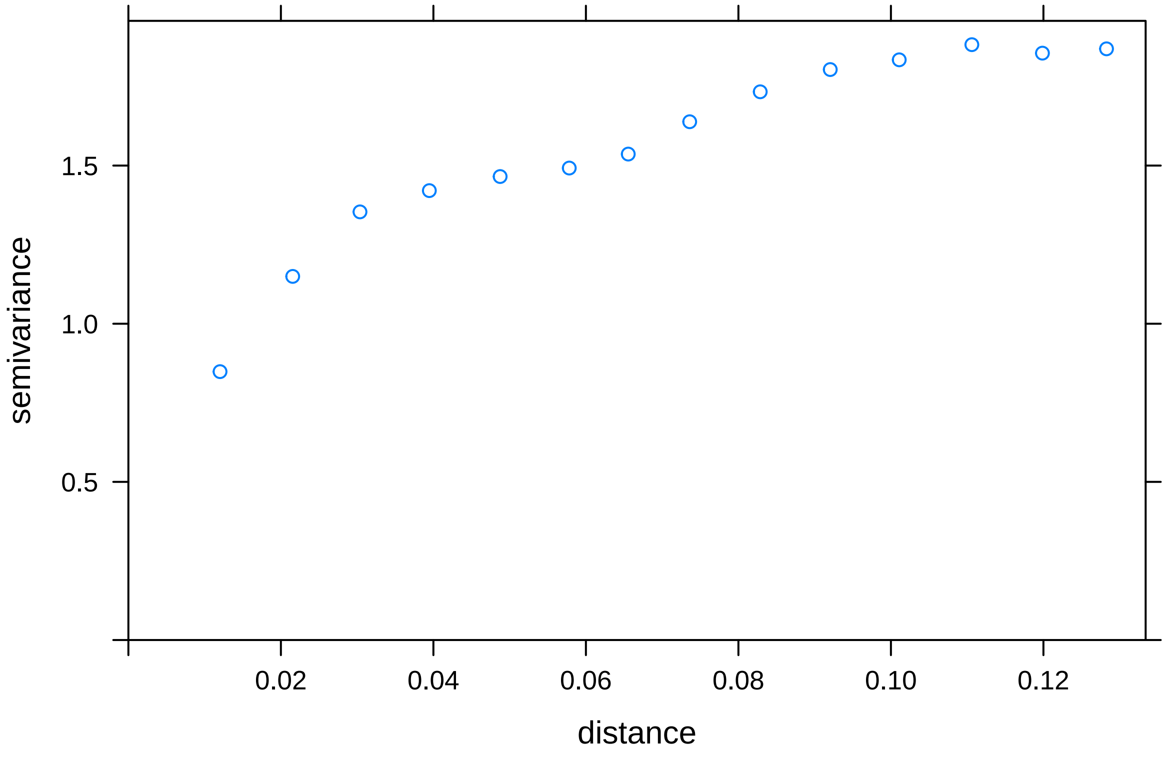
3. Fit a `lm()` of temperature by land cover type and plot a map of the residuals to convince yourself that the residuals are correlated.

```
temp_lm <- lm(Temp ~ Surface, data)
resids <- temp_lm$residuals
newdf1 <- data %>% filter(!is.na(Temp))
newdf2 <- cbind(newdf1, resids)
ggplot(data = newdf2, mapping = aes(x = Lon, y = Lat, fill = resids)) +
  geom_raster() +
  scale_fill_distiller(palette = "Spectral", na.value = NA) +
  labs(title = "Surface Temperature Residuals in Houston, Texas",
       x = "Longitude",
       y = "Latitude")
```



4. Draw a variogram of the residuals from an `lm()` fit of temperature to land cover type.

```
myVariogram <- variogram(object = Temp ~ Surface, locations = ~ Lon + Lat, data = newdf1)
plot(myVariogram)
```



Spatial MLR Model Fitting

1. Choose (with AIC) among an exponential, spherical and Gaussian correlation structure (with nuggets) by fitting each model to `Temperature` using land cover as a factor covariate. For the best fit model, identify the *constrained* estimates of the correlation structure along with any $\hat{\beta}$ coefficients and the estimate of the variance parameter σ^2.

```
model1 <- gls(model = Temp~Surface, data = newdf1,
              correlation = corExp(form=~Lon+Lat, nugget=TRUE), method = "ML")
model2 <- gls(model = Temp~Surface, data = newdf1,
              correlation = corSpher(form=~Lon+Lat, nugget=TRUE), method = "ML")
model3 <- gls(model = Temp~Surface, data = newdf1,
              correlation = corGaus(form=~Lon+Lat, nugget=TRUE), method = "ML") # This one is best
AIC(model1)
```

```
## [1] 1191.253
```

```
AIC(model2)
```

```
## [1] 1189.086
```

```
AIC(model3)
```

```
## [1] 1188.856
```

```r
#coef(model1$modelStruct$corStruct, unconstrained=FALSE)
#coef(model2$modelStruct$corStruct, unconstrained=FALSE)
coef(model3$modelStruct$corStruct, unconstrained=FALSE)
```

```
##      range     nugget 
## 0.02863235 0.04134444
```

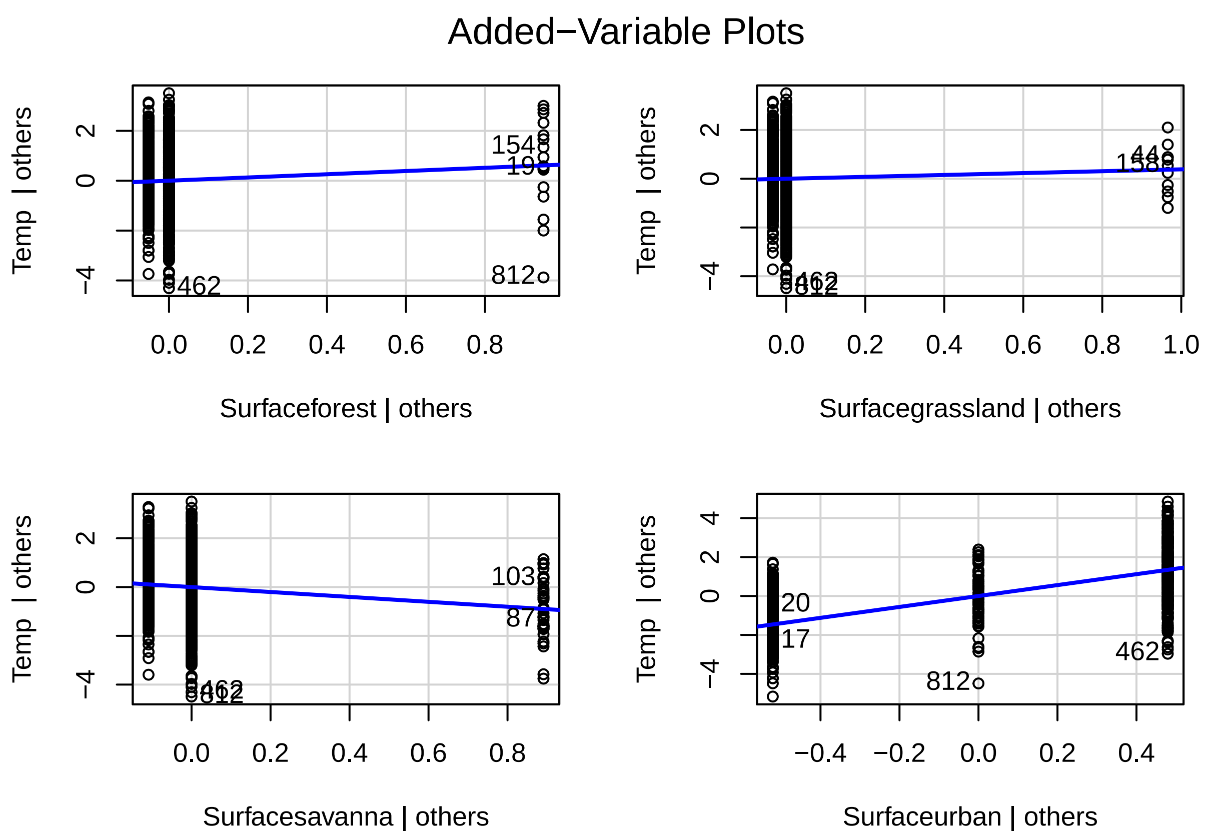
```r
summary(model3)
```

```
## Generalized least squares fit by maximum likelihood
##   Model: Temp ~ Surface 
##   Data: newdf1 
##        AIC      BIC    logLik
##   1188.856 1225.435 -586.4282
## 
## Correlation Structure: Gaussian spatial correlation
##  Formula: ~Lon + Lat 
##  Parameter estimate(s):
##      range     nugget 
## 0.02863235 0.04134444 
## 
## Coefficients:
##                     Value  Std.Error   t-value p-value
## (Intercept)      34.06176 0.25357007 134.32878  0.0000
## Surfaceforest    -0.08052 0.12182122  -0.66098  0.5088
## Surfacegrassland -0.14724 0.12326451  -1.19453  0.2327
## Surfacesavanna   -0.13270 0.08009510  -1.65683  0.0980
## Surfaceurban      0.18836 0.06047367   3.11467  0.0019
## 
##  Correlation: 
##                  (Intr) Srfcfr Srfcgr Srfcsv
## Surfaceforest    -0.031                     
## Surfacegrassland -0.015  0.007              
## Surfacesavanna   -0.051  0.126  0.053       
## Surfaceurban     -0.129  0.130  0.091  0.142
## 
## Standardized residuals:
##         Min          Q1         Med          Q3         Max 
## -2.95367620 -0.86814779 -0.06006239  0.93370933  3.04380785 
## 
## Residual standard error: 1.649212 
## Degrees of freedom: 715 total; 710 residual
```

Validating Spatial MLR Model Assumptions and Predictions

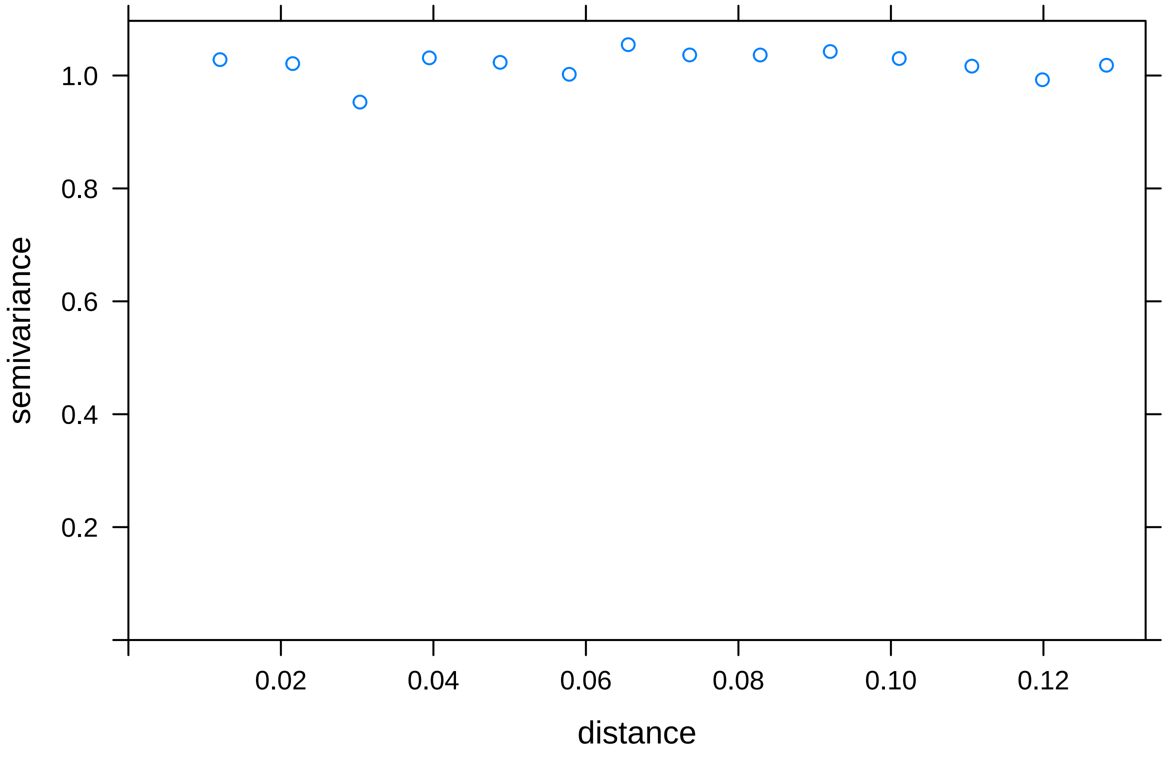
1. Check the assumption of linearity using added-variable plots from an independent model (note: correlation doesn't change the linearity at all so you can just fit an independent model and look at the added-variable plots).

```r
avPlots(temp_lm, ask = FALSE)
```



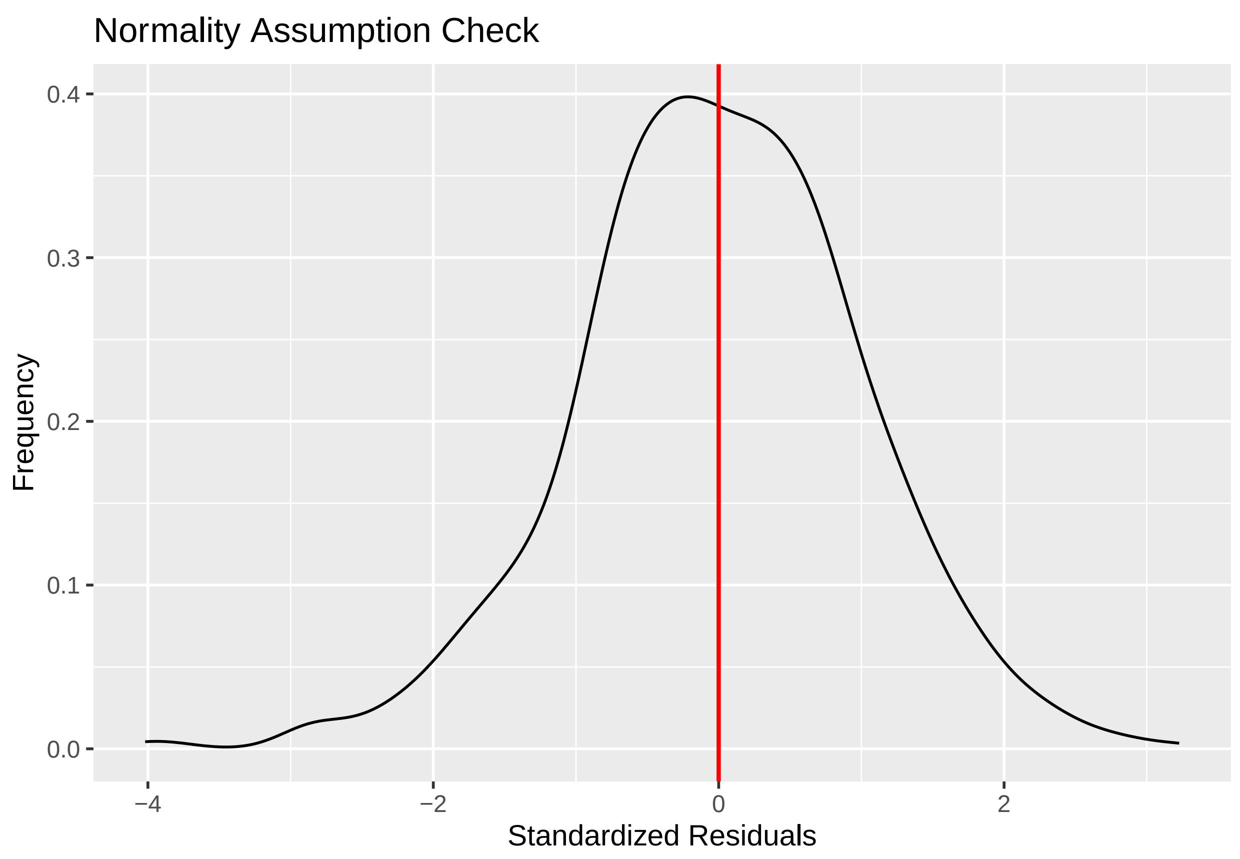
2. Check the assumption of independence by decorrelating residuals and looking at the variogram of decorrelated residuals.

```
sres <- stdres.gls(model3)
residDF <- data.frame(Lon=newdf1$Lon, Lat=newdf1$Lat, decorrResid=sres)
residVariogram <- variogram(object=decorrResid~1, locations=~Lon+Lat, data=residDF)
plot(residVariogram)
```



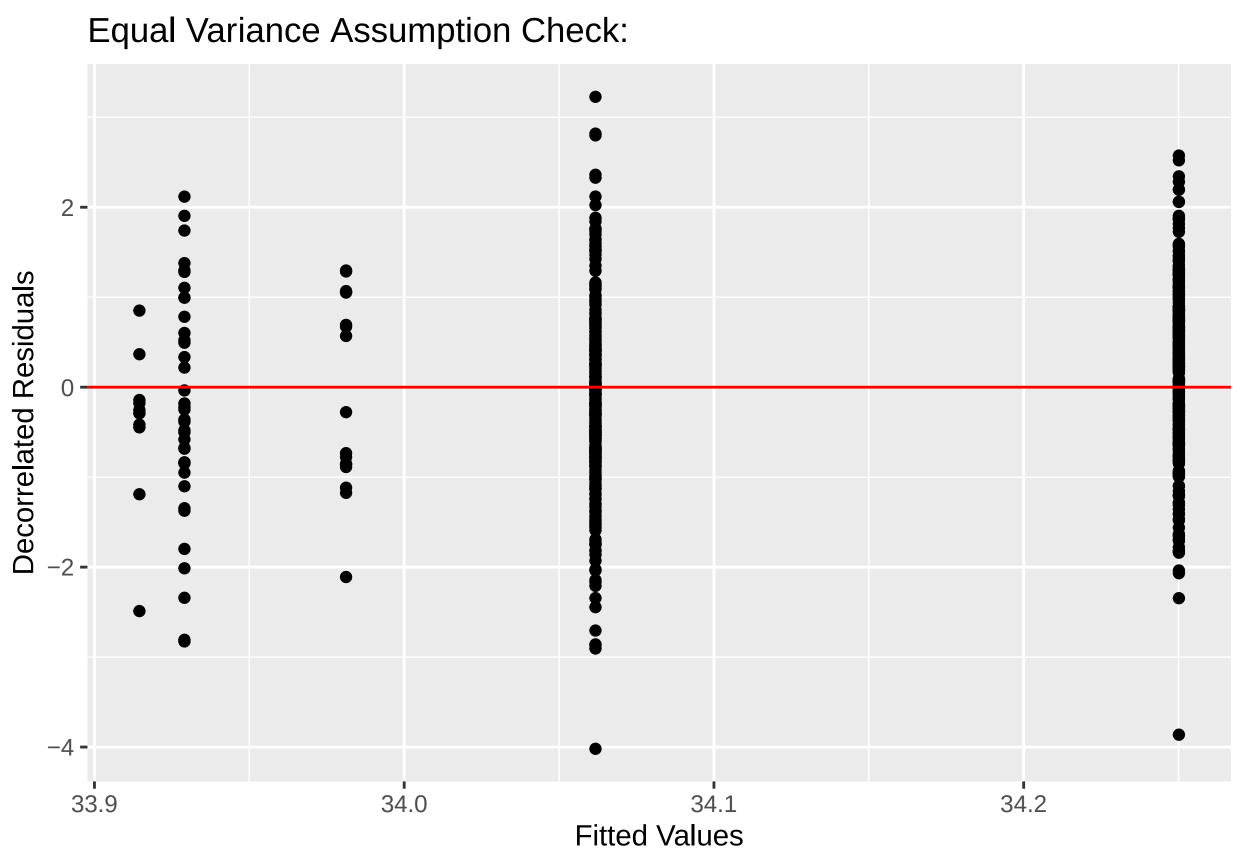
3. Check the assumption of normality by drawing a histogram of the decorrelated residuals.

```
ggplot() +
  geom_density(mapping = aes(x = sres)) + #we could change log to base=>1, but it won't center over 0
  xlab('Standardized Residuals') +
  ylab('Frequency') +
  ggtitle('Normality Assumption Check') +
  geom_vline(xintercept = 0, col = "red", linewidth = 0.75)
```



4. Check the assumption of equal variance by plotting the decorrelated residuals vs. the fitted values.

```
ggplot(mapping = aes(x=model3$fitted, y=sres)) +
  geom_point() +
  xlab('Fitted Values') +
  ylab('Decorrelated Residuals') +
  ggtitle('Equal Variance Assumption Check:') +
  geom_hline(yintercept = 0, col = "red")
```



5. Clock how long it takes `gls()` to fit the spatial model with `system.time()`.

```
system.time({
  gls(model = Temp~Surface, data = newdf1,
               correlation = corGaus(form=~Lon+Lat, nugget=TRUE), method = "ML")
})
```

```
##    user  system elapsed
##   28.56    0.39   28.97
```

6. Run 50 CV studies to assess the predictive accuracy of your model in terms of bias, RPMSE, coverage and width and use a progress bar to track the progress of the loop. Compare these estimates to the predictions under an uncorrelated model with `lm()`.

```
set.seed(42) #for reproducibility
n.cv <- 50 #Number of CV studies to run
n.test <- 143 #Number of observations in a test set
# n.test = 143 is about 20% of 715
pb <- txtProgressBar(min = 0, max = n.cv, style = 3)
```

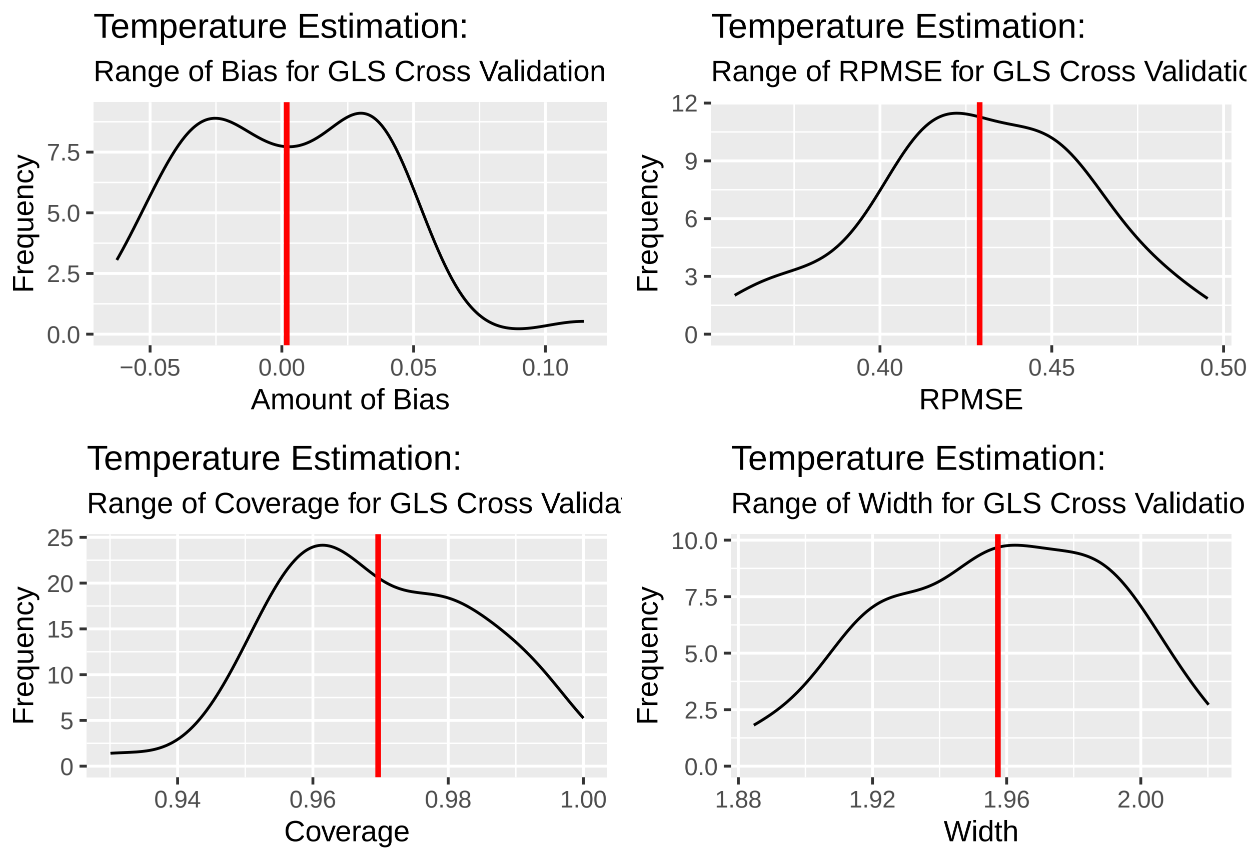
```
##   |                                                                                |
```

```r
rpmse <- rep(x=NA, times=n.cv)
bias <- rep(x=NA, times=n.cv)
wid <- rep(x=NA, times=n.cv)
cvg <- rep(x=NA, times=n.cv)
for(cv in 1:n.cv){
  ## Select test observations
  test.obs <- sample(x=1:nrow(newdf1), size=n.test)

  ## Split into test and training sets
  test.set <- newdf1[test.obs,]
  train.set <- newdf1[-test.obs,]

  ## Fit a lm() using the training data
  train.lm <- gls(model = Temp~Surface, data = train.set,
             correlation = corGaus(form=~Lon+Lat, nugget=TRUE), method = "ML")

  ## Generate predictions for the test set
  my.preds <- predictgls(train.lm, newdframe = test.set)

  ## Calculate bias
  bias[cv] <- mean(my.preds[,'Prediction']-test.set[['Temp']])

  ## Calculate RPMSE
  rpmse[cv] <- (test.set[['Temp']]-my.preds[,'Prediction'])^2 %>% mean() %>% sqrt()

  ## Calculate Coverage
  cvg[cv] <- ((test.set[['Temp']] > my.preds[,'lwr']) &
               (test.set[['Temp']] < my.preds[,'upr'])) %>% mean()

  ## Calculate Width
  wid[cv] <- (my.preds[,'upr'] - my.preds[,'lwr']) %>% mean()

  ## Update the progress bar
  setTxtProgressBar(pb, cv)
}
```

```
##   |                                                                              |=
```

```r
close(pb)
```

```r
CV.bias <- ggplot() +
  geom_density(mapping=aes(x=bias)) +
  xlab('Amount of Bias') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of Bias for GLS Cross Validation') +
  geom_vline(xintercept = mean(bias), col = "red", lwd = 1)

CV.RPMSE <- ggplot() +
  geom_density(mapping=aes(x=rpmse)) +
  xlab('RPMSE') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of RPMSE for GLS Cross Validation') +
  geom_vline(xintercept = mean(rpmse), col = "red", lwd = 1)

CV.coverage <- ggplot() +
  geom_density(mapping=aes(x=cvg)) +
  xlab('Coverage') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of Coverage for GLS Cross Validation') +
  geom_vline(xintercept = mean(cvg), col = "red", lwd = 1)

CV.width <- ggplot() +
  geom_density(mapping=aes(x=wid)) +
  xlab('Width') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of Width for GLS Cross Validation') +
  geom_vline(xintercept = mean(wid), col = "red", lwd = 1)

suppressMessages(grid.arrange(CV.bias, CV.RPMSE, CV.coverage, CV.width, nrow=2))
```



```
print(paste("Mean GLS bias:",mean(bias)))
```

```
## [1] "Mean GLS bias: 0.00182265982019386"
```

```r
print(paste("Mean GLS RPMSE:",mean(rpmse)))
```

```
## [1] "Mean GLS RPMSE: 0.429002828821189"
```

```r
print(paste("Mean GLS coverage:",mean(cvg)))
```

```
## [1] "Mean GLS coverage: 0.96965034965035"
```

```r
print(paste("Mean GLS width:", mean(wid)))
```

```
## [1] "Mean GLS width: 1.95738542504373"
```

```r
print(paste("Standard deviation of data:",sd(newdf1$Temp)))
```

```
## [1] "Standard deviation of data: 1.99581885059004"
```

```r
set.seed(42) #for reproducibility
# n.test = 143 is about 20% of 715
pb2 <- txtProgressBar(min = 0, max = n.cv, style = 3)
```

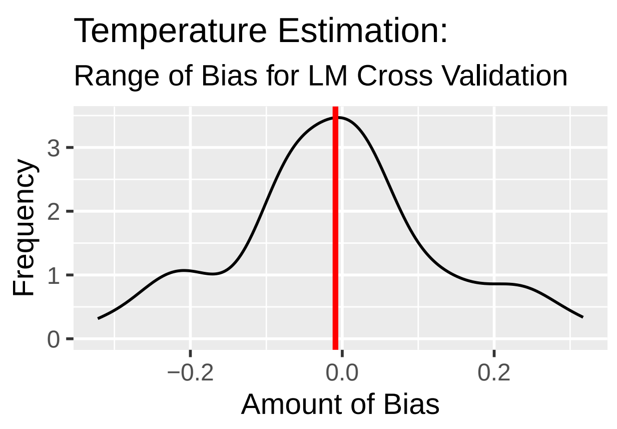
```
##   |                                                                              |
```

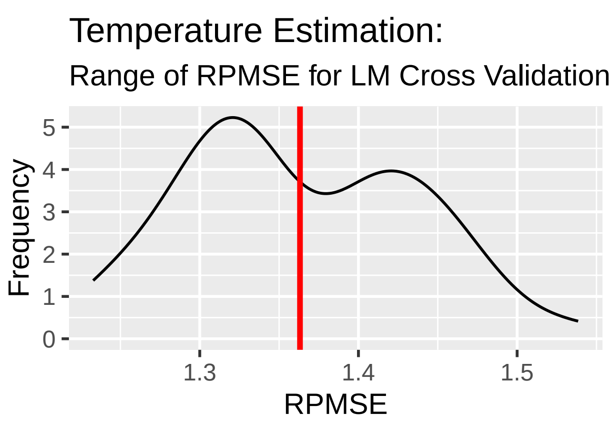
```r
rpmse2 <- rep(x=NA, times=n.cv)
bias2 <- rep(x=NA, times=n.cv)
wid2 <- rep(x=NA, times=n.cv)
cvg2 <- rep(x=NA, times=n.cv)
for(cv in 1:n.cv){
  ## Select test observations
  test.obs2 <- sample(x=1:nrow(newdf1), size=n.test)

  ## Split into test and training sets
  test.set2 <- newdf1[test.obs2,]
  train.set2 <- newdf1[-test.obs2,]

  ## Fit a lm() using the training data
  train.lm2 <- lm(Temp ~ Surface, train.set2)

  ## Generate predictions for the test set
  my.preds2 <- predict.lm(train.lm2, newdata=test.set2, interval="prediction")

  bias2[cv] <- mean(my.preds2[,1]- test.set2$Temp)
  rpmse2[cv] <- sqrt(mean((my.preds2[,1] - test.set2$Temp)^2))
  cvg2[cv] <- mean((my.preds2[,2] < test.set2$Temp) &
                      (my.preds2[,3] > test.set2$Temp))
  wid2[cv] <- mean(my.preds2[,3] - my.preds2[,2])

  ## Update the progress bar
  setTxtProgressBar(pb2, cv)
}
```

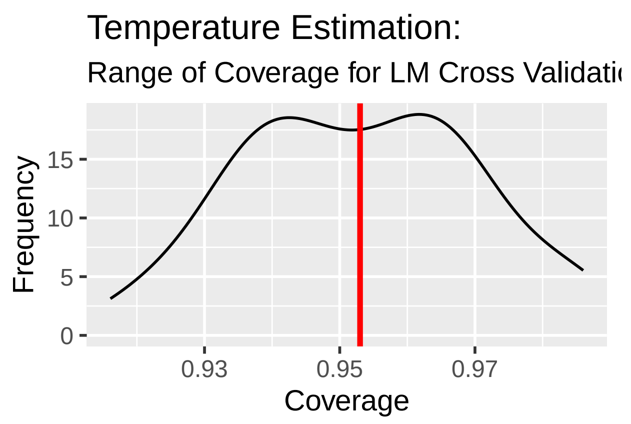
```
##   |                                                                                  |=
```

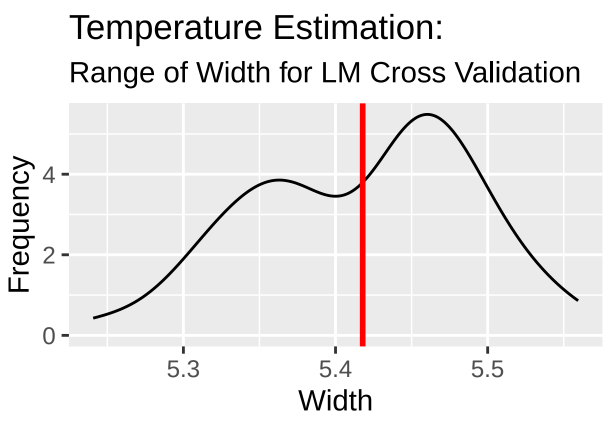
```
close(pb2)
```

```
CV.bias2 <- ggplot() +
  geom_density(mapping=aes(x=bias2)) +
  xlab('Amount of Bias') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of Bias for LM Cross Validation') +
  geom_vline(xintercept = mean(bias2), col = "red", lwd = 1)

CV.RPMSE2 <- ggplot() +
  geom_density(mapping=aes(x=rpmse2)) +
  xlab('RPMSE') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of RPMSE for LM Cross Validation') +
  geom_vline(xintercept = mean(rpmse2), col = "red", lwd = 1)

CV.coverage2 <- ggplot() +
  geom_density(mapping=aes(x=cvg2)) +
  xlab('Coverage') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of Coverage for LM Cross Validation') +
  geom_vline(xintercept = mean(cvg2), col = "red", lwd = 1)

CV.width2 <- ggplot() +
  geom_density(mapping=aes(x=wid2)) +
  xlab('Width') +
  ylab('Frequency') +
  ggtitle('Temperature Estimation:',
          subtitle = 'Range of Width for LM Cross Validation') +
  geom_vline(xintercept = mean(wid2), col = "red", lwd = 1)

suppressMessages(grid.arrange(CV.bias2, CV.RPMSE2, CV.coverage2, CV.width2, nrow=2))
```









```r
print(paste("Mean LM bias:",mean(bias2)))
```

```
## [1] "Mean LM bias: -0.00898083479472803"
```

```r
print(paste("Mean LM RPMSE:",mean(rpmse2)))
```

```
## [1] "Mean LM RPMSE: 1.36316020539852"
```

```r
print(paste("Mean LM coverage:",mean(cvg2)))
```

```
## [1] "Mean LM coverage: 0.953006993006993"
```

```r
print(paste("Mean LM width:", mean(wid2)))
```

```
## [1] "Mean LM width: 5.41784573485543"
```

```r
print(paste("Standard deviation of data:",sd(newdf1$Temp)))
```

```
## [1] "Standard deviation of data: 1.99581885059004"
```

Statistical Inference

1. Use an F-test to see if temperatures are difference across any of the land-cover types.

```
anova(model3)
```

```
## Denom. DF: 710
##             numDF   F-value p-value
## (Intercept)     1 18460.730  <.0001
## Surface         4     4.228  0.0022
```

```
#reject H_0, land cover has an effect
```

$$H_0 : \beta_{forest} = \beta_{grassland} = \beta_{savanna} = \beta_{urban} = 0$$
$$H_A : \text{at least one } \beta \text{ is non-zero}$$

The p-value of the F-test is 0.0022, so we reject the null hypothesis and conclude there is evidence that temperatures are different across land-cover types.

2. Create confidence intervals for each effect of land cover and determine which land cover types result in increased temperatures.

```
confint(model3)
```

```
##                         2.5 %      97.5 %
## (Intercept)      33.56477163 34.55874806
## Surfaceforest    -0.31928653  0.15824390
## Surfacegrassland -0.38883755  0.09435045
## Surfacesavanna   -0.28968774  0.02427929
## Surfaceurban      0.06982921  0.30688163
```

```
#intervals(model3)[1]
```

The land cover type of urban results in increased temperatures.

3. Perform a GLHT to construct a confidence interval of the difference temperature between Savannah and Urban land covers.

```
sav = c(1, 0, 0, 1, 0)
urb = c(1, 0, 0, 0, 1)

mytest <- glht(model3, linfct = t(sav-urb))
summary(mytest)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: gls(model = Temp ~ Surface, data = newdf1, correlation = corGaus(form = ~Lon +
##     Lat, nugget = TRUE), method = "ML")
```

```
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0 -0.32106    0.09323  -3.444 0.000574 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

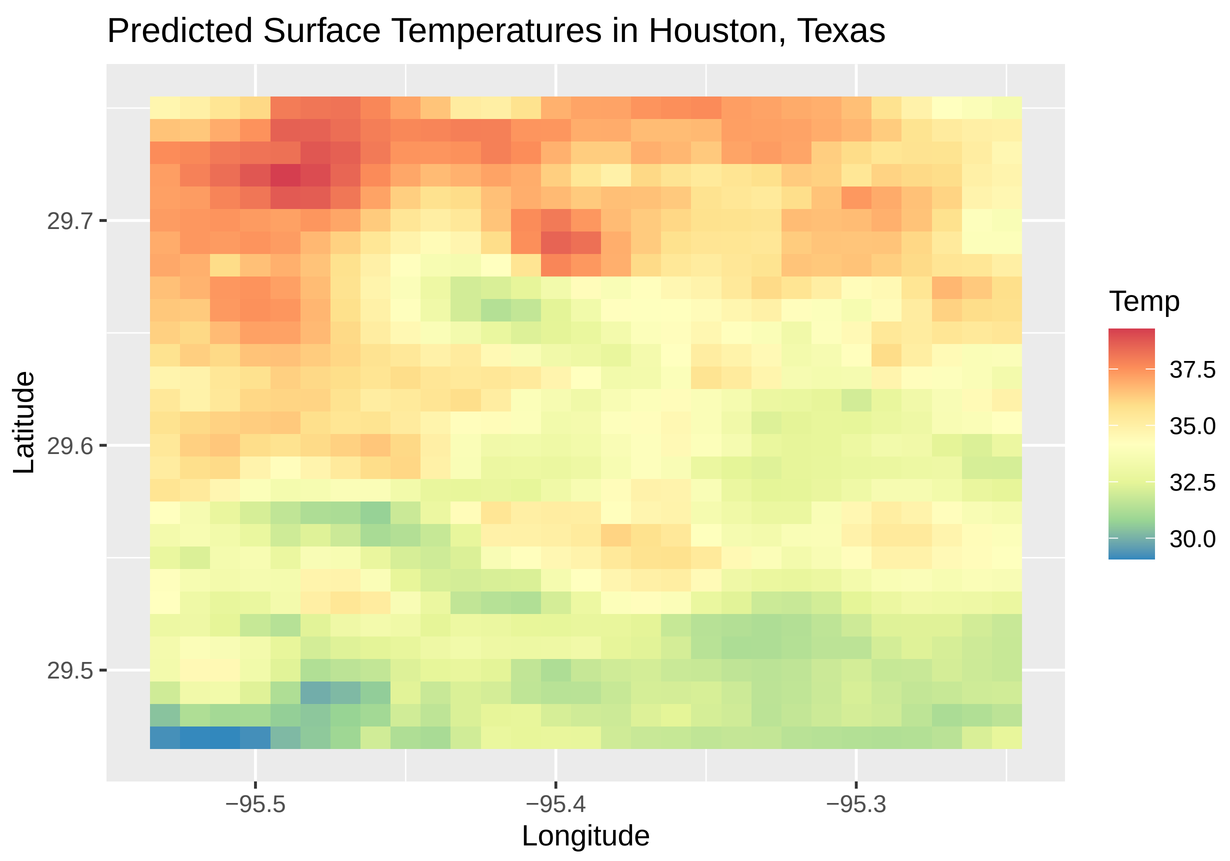
```
confint(mytest)
```

```
##
##   Simultaneous Confidence Intervals
##
## Fit: gls(model = Temp ~ Surface, data = newdf1, correlation = corGaus(form = ~Lon +
##     Lat, nugget = TRUE), method = "ML")
##
## Quantile = 1.96
## 95% family-wise confidence level
##
##
## Linear Hypotheses:
##        Estimate lwr     upr
## 1 == 0 -0.3211  -0.5038 -0.1383
```

I am 95% confident that the difference in temperatures between Savannah and Urban land covers is between 0.1383 and 0.5038 degrees.

4. Create and map predictions of the temperature at each location that was impeded by cloud cover.

```
newdf3 <- data %>% filter(is.na(Temp))
dataPreds <- predictgls(glsobj=model3, newdframe=newdf3)
newdf4 <- dataPreds %>% reframe(Lon, Lat, Prediction, Surface)
newdf4 <- rename(newdf4, Temp = Prediction)
data2 <- rbind(newdf1, newdf4)

ggplot(data = data2, mapping = aes(x = Lon, y = Lat, fill = Temp)) +
  geom_raster() +
  scale_fill_distiller(palette = "Spectral", na.value = NA) +
  labs(title = "Predicted Surface Temperatures in Houston, Texas",
       x = "Longitude",
       y = "Latitude")
```


Predicted Surface Temperatures in Houston, Texas
Latitude
29.7
29.6
29.5
−95.5
−95.4
−95.3
Longitude
Temp
37.5
35.0
32.5
30.0