C173 Final Project (ArcGIS Data)

Daren Sathasivam

2025-03-14

NASA FIRMS Wildfire data using MODIS Algorithm for Fire Detection

```
#https://firms.modaps.eosdis.nasa.gov/download/list.php
# 1-6-2025 to 1-24-2025
data <- read.csv("FIRMS_MODIS.csv")
dim(data)
```

```
## [1] 2572   14
```

```
str(data)
```

```
## 'data.frame':    2572 obs. of  14 variables:
##  $ latitude  : num  19.4 20.9 19.4 19.4 19.4 ...
##  $ longitude : num  -155 -156 -155 -155 -155 ...
##  $ brightness: num  396 315 344 351 357 ...
##  $ scan      : num  1 1.04 1 1 1 1.72 1.72 1.72 1.73 1.73 ...
##  $ track     : num  1 1.02 1 1 1 1.29 1.29 1.29 1.29 1.29 ...
##  $ acq_date  : chr  "2025-01-06" "2025-01-06" "2025-01-06" "2025-01-06" ...
##  $ acq_time  : int  34 34 34 34 34 459 459 459 459 459 ...
##  $ satellite : chr  "Aqua" "Aqua" "Aqua" "Aqua" ...
##  $ instrument: chr  "MODIS" "MODIS" "MODIS" "MODIS" ...
##  $ confidence: int  100 41 87 95 90 92 100 93 100 70 ...
##  $ version   : chr  "6.1NRT" "6.1NRT" "6.1NRT" "6.1NRT" ...
##  $ bright_t31: num  339 303 310 315 321 ...
##  $ frp       : num  219.67 5.52 39.97 53.45 69.03 ...
##  $ daynight  : chr  "D" "D" "D" "D" ...
```

Clean Data

```
# Load libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```
library(geoR)
```

```
## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.9-4 (built on 2024-02-14) is now loaded
## ----------------------------------------------------------
```

```
library(sp)
library(gstat)
library(ggplot2)
library(tidyr)
library(e1071)

# --- Clean data ---
ca_data <- data %>%
  filter(latitude >= 32.5 & latitude <= 42,
         longitude >= -124.4 & longitude <= -114.13)
cal_data <- ca_data %>%
  mutate(acq_date = as.Date(acq_date))
# Select variables
final_data <- ca_data %>%
  select(longitude, latitude, frp, brightness, bright_t31, confidence, scan, daynight, acq_date)
# Order data by date and remove duplicates
final_data <- final_data %>%
  arrange(acq_date) %>%
  distinct(latitude, longitude, .keep_all = TRUE)
final_data$daynight <- as.factor(final_data$daynight)
final_data$acq_date <- as.Date(final_data$acq_date)

dim(final_data)
```

```
## [1] 755   9
```

```
str(final_data)
```

```
## 'data.frame':    755 obs. of  9 variables:
##  $ longitude : num  -121 -121 -120 -121 -121 ...
##  $ latitude  : num  36.2 36.1 35.6 36.1 36.4 ...
##  $ frp       : num  28.45 20.23 6.31 22.73 5.39 ...
##  $ brightness: num  303 300 306 325 304 ...
##  $ bright_t31: num  286 286 284 293 292 ...
##  $ confidence: int  53 13 62 83 57 68 62 45 76 86 ...
##  $ scan      : num  2.75 2.75 1.01 1.03 1.04 1.04 1.04 1.01 1.03 1.03 ...
##  $ daynight  : Factor w/ 2 levels "D","N": 1 1 1 1 1 1 1 1 1 1 ...
##  $ acq_date  : Date, format: "2025-01-06" "2025-01-06" ...
```

```
# range(final_data$acq_date)
colSums(is.na(final_data))
```

```
##  longitude   latitude        frp brightness bright_t31 confidence       scan
##          0          0          0          0          0          0          0
##   daynight   acq_date
##          0          0
```

Dataset Description

Dataset Dimensions:

- Rows: 755 Observations
- Cols: 9 Columns

Coordinates:

- 1. *longitude*
  - Description: Longitude of the incident's reported point of origin.
  - Type: Numeric (decimal degrees).
  - Relevance: Spatial reference(x) that helps identify fire locations in relation to California's geography
- 2. *latitude*
  - Description: Latitude of the incident's reported point of origin.
  - Type: Numeric (decimal degrees).
  - Relevance: Spatial reference(y) that helps identify fire locations in relation to California's geography

Target Variable:

- 3. *frp*
  - Description: Fire Radiative Power/Energy output from fires
  - Type: Numeric.
  - Relevance: Direct measure of fire intensity/energy.

Predictor Variables:

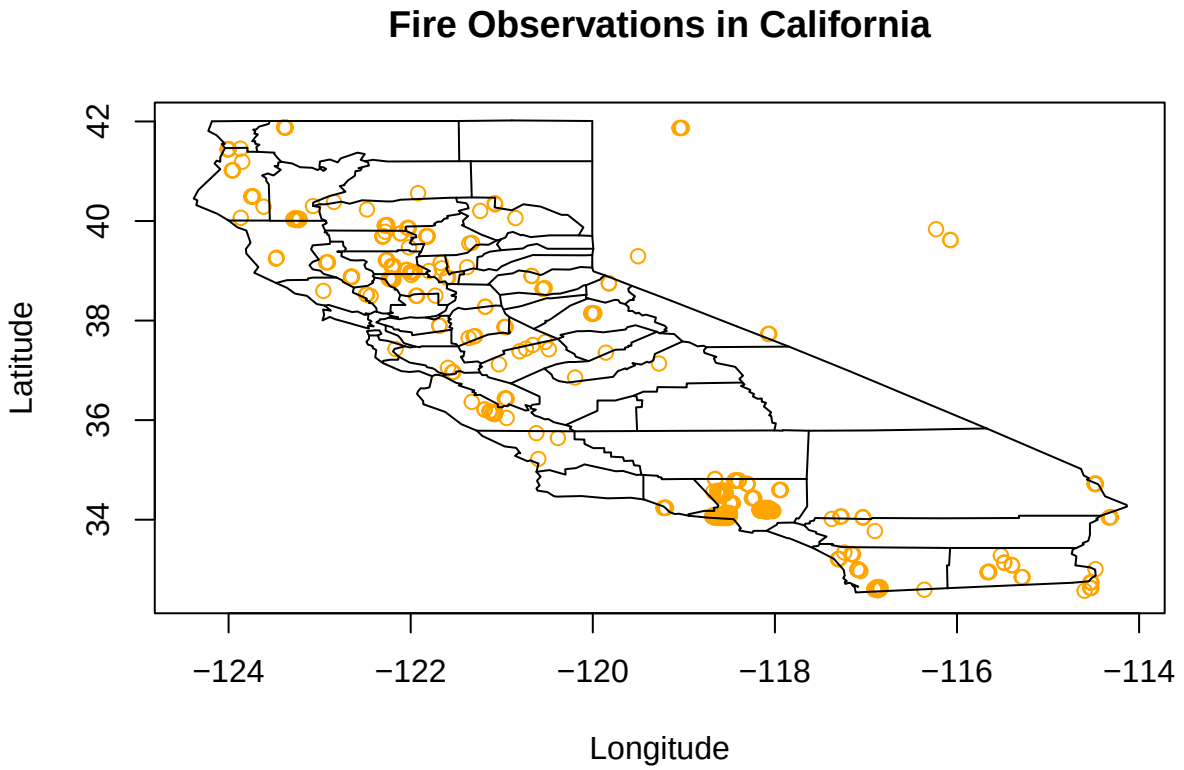
- 4. *brightness*
  - Description:Brightness temperature - Thermal radiation at fire location.
  - Type: Numeric (Kelvin).
  - Relevance: Higher values indicate stronger thermal signals from fires.
- 5. **bright_t31**
  - Description: Brightness temperature measured in MODIS band 31.
  - Type: Numeric(Kelvin).
  - Relevance: Another thermal measure capturing different infrared wavelengths of the fire.
- 6. **confidence**
  - Description: Detection confidence - Reliability of fire detection
  - Type: Integer(0-100).
  - Relevance: Higher confidence readings may correlate with more accurate FRP values.
- 7. **scan**
  - Description: Approximate pixel size on the ground in the cross-track direction.
  - Type: Numeric(km).
  - Relevance: Larger scan areas may influence fire detection characteristics.
- 8. **daynight*
  - Description: Day/Night flag.
  - Type: Categorical (2 levels - "D"/"N").
  - Relevance: Fire detection and thermal readings can differ by day vs. night conditions.

- 9. **acq_date**
  - Description: Acquisition date.
  - Type: Character (YYYY-MM-DD).
  - Relevance: Organizational variable - Not for modeling, but useful for temporal sorting/analysis.

In this analysis, my target variable is IncidentSize. By doing this, the model can predict the size of a fire based on the predictors, DiscoveryAcres, IncidentTypeCategory, FireCause, POOProtectingAgency, GACC, and POOLandownerKind, which provide continuous and categorical information that may help explain variations in the final fire size. The coordinate values allow for spatial interpolation and geostatistical modeling.

Exploratory Data Analysis(EDA)

```
library(maps)
plot(x = final_data$longitude, y = final_data$latitude,
     col = "orange",
     xlim = c(-124.4, -114.13),
     ylim = c(32.5, 42),
     xlab = "Longitude", ylab = "Latitude", main = "Fire Observations in California")
map("county", "California", add = TRUE)
```




```
# Histograms, ecdfs, scatterplots, descriptive statistics, etc
numeric_vars <- c("frp", "brightness", "bright_t31", "confidence", "scan")
cat_var <- "daynight"

# Summaries for numeric
summary_stats <- final_data %>%
  select(all_of(numeric_vars)) %>%
  summary()
summary_stats
```

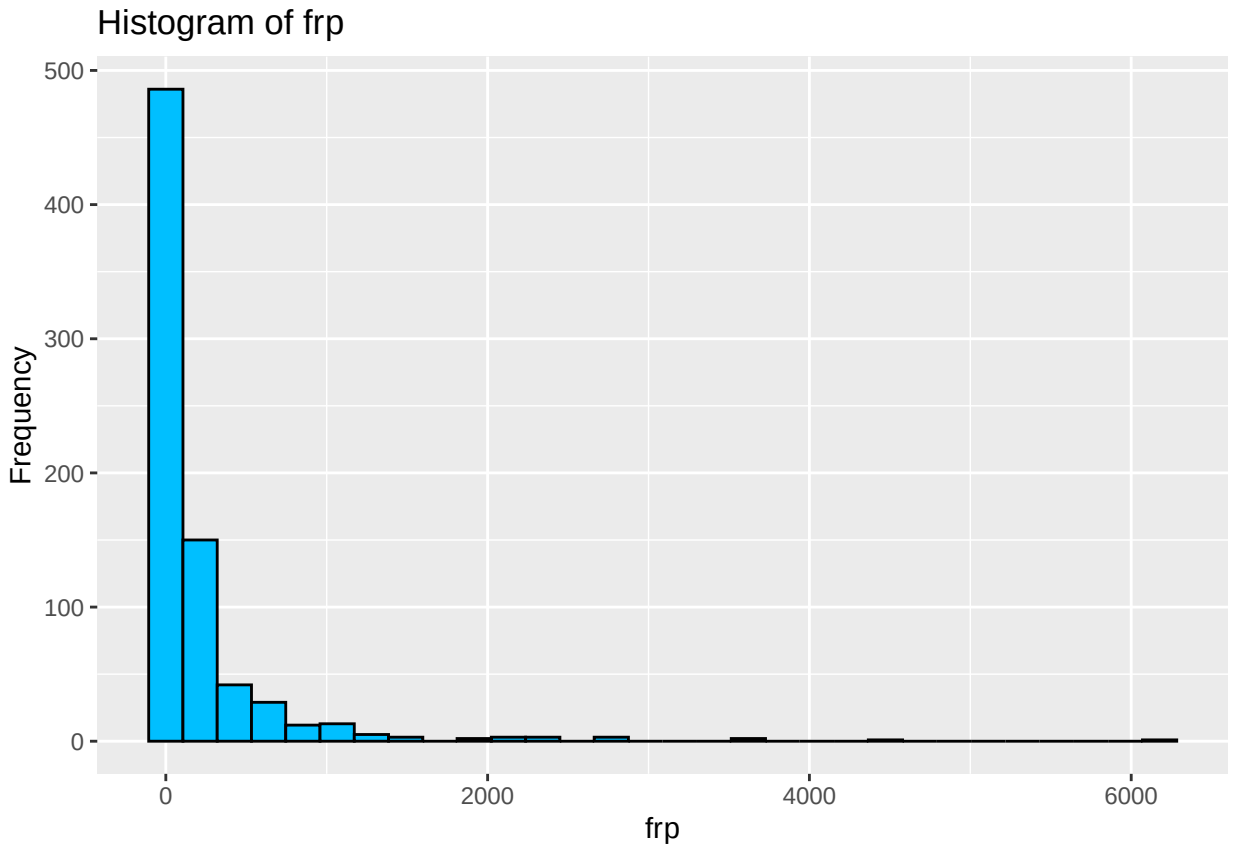
```
##       frp              brightness      bright_t31      confidence
##  Min.   :   0.00   Min.   :300.0   Min.   :268.7   Min.   :  0.00
##  1st Qu.:  15.12   1st Qu.:308.7   1st Qu.:285.9   1st Qu.: 63.00
##  Median :  51.81   Median :329.2   Median :290.1   Median : 87.00
##  Mean   : 208.40   Mean   :340.9   Mean   :292.0   Mean   : 78.41
##  3rd Qu.: 189.74   3rd Qu.:361.9   3rd Qu.:295.7   3rd Qu.:100.00
##  Max.   :6176.41   Max.   :497.4   Max.   :400.1   Max.   :100.00
##       scan
##  Min.   :1.000
##  1st Qu.:1.080
```

```
##  Median :1.340
##  Mean   :1.659
##  3rd Qu.:1.830
##  Max.   :4.220
```

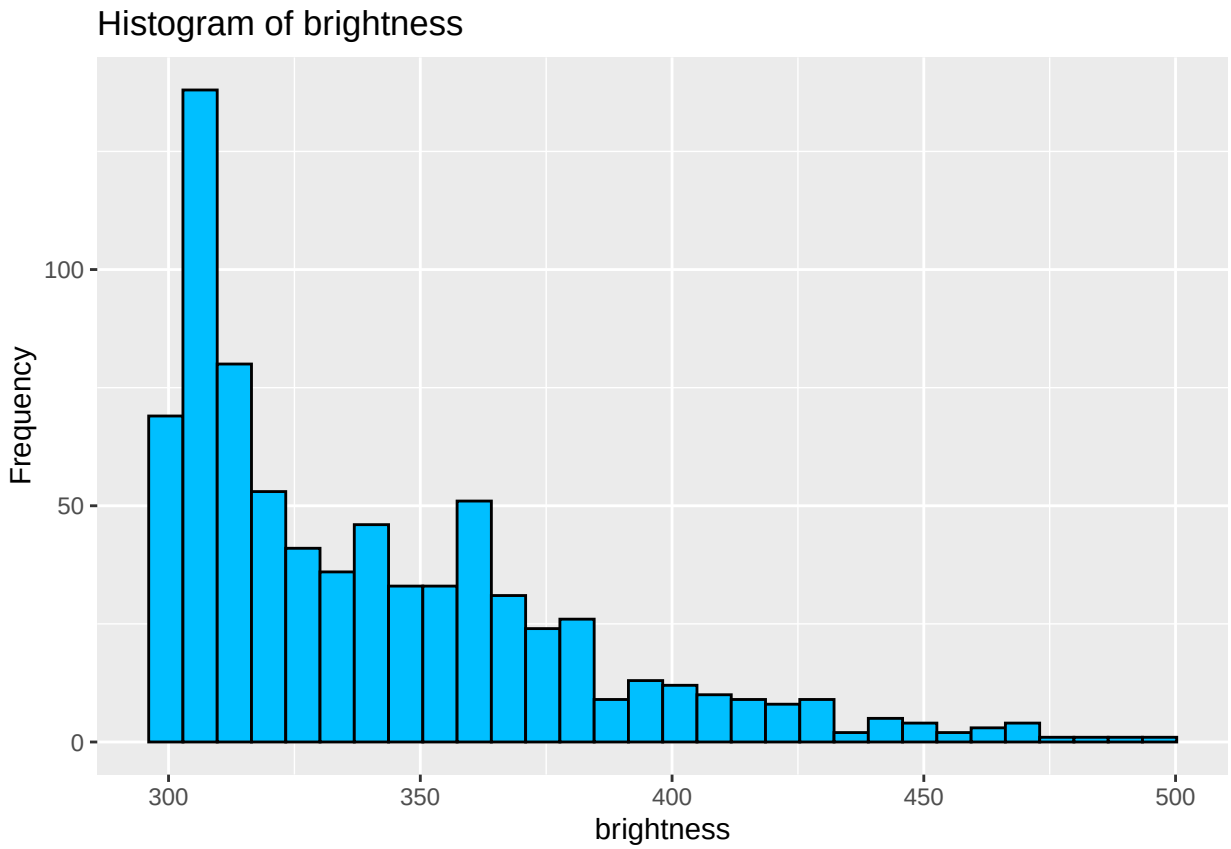
```r
par(mfrow = c(2, 2))
# Histograms for numeric
for (var in numeric_vars) {
  p <- ggplot(final_data, aes_string(x = var)) +
    geom_histogram(fill = "deepskyblue", col = "black") +
    ggtitle(paste("Histogram of", var)) +
    xlab(var) +
    ylab("Frequency")
  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

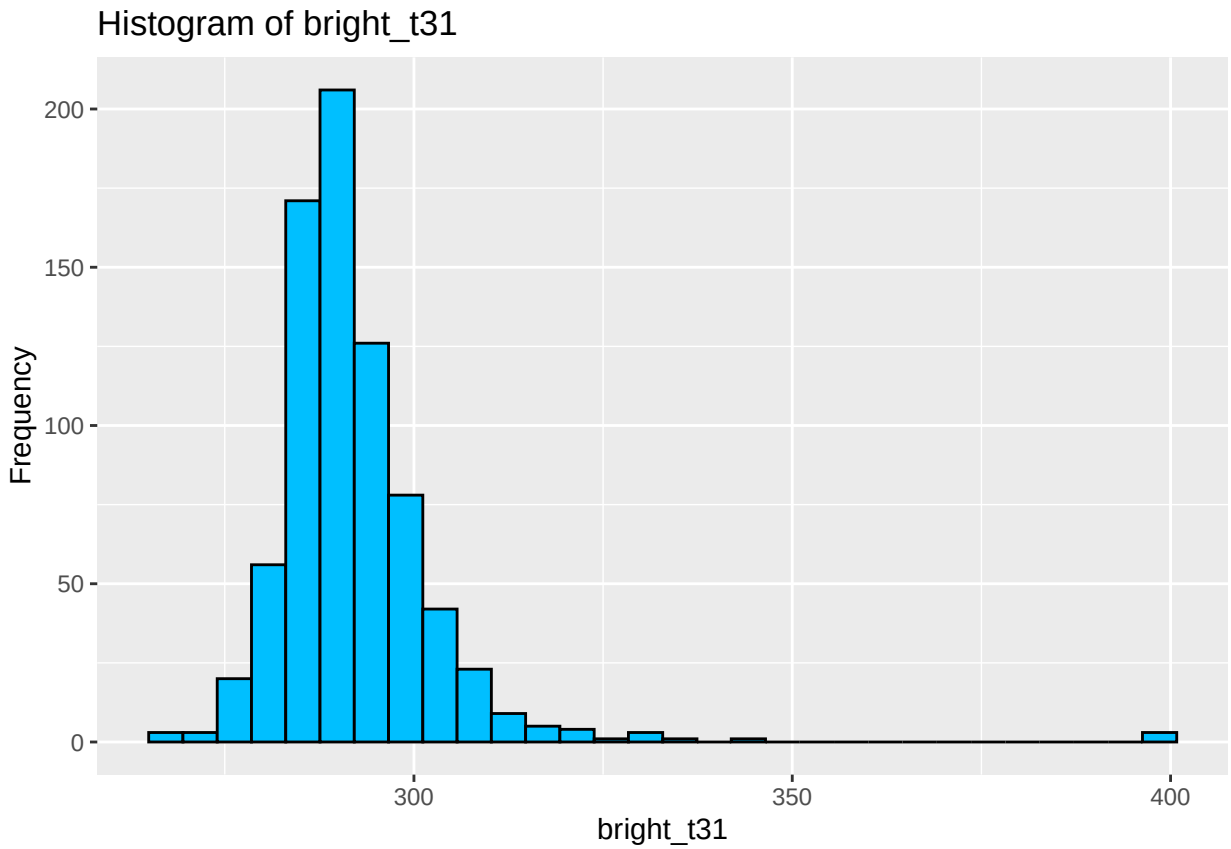
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



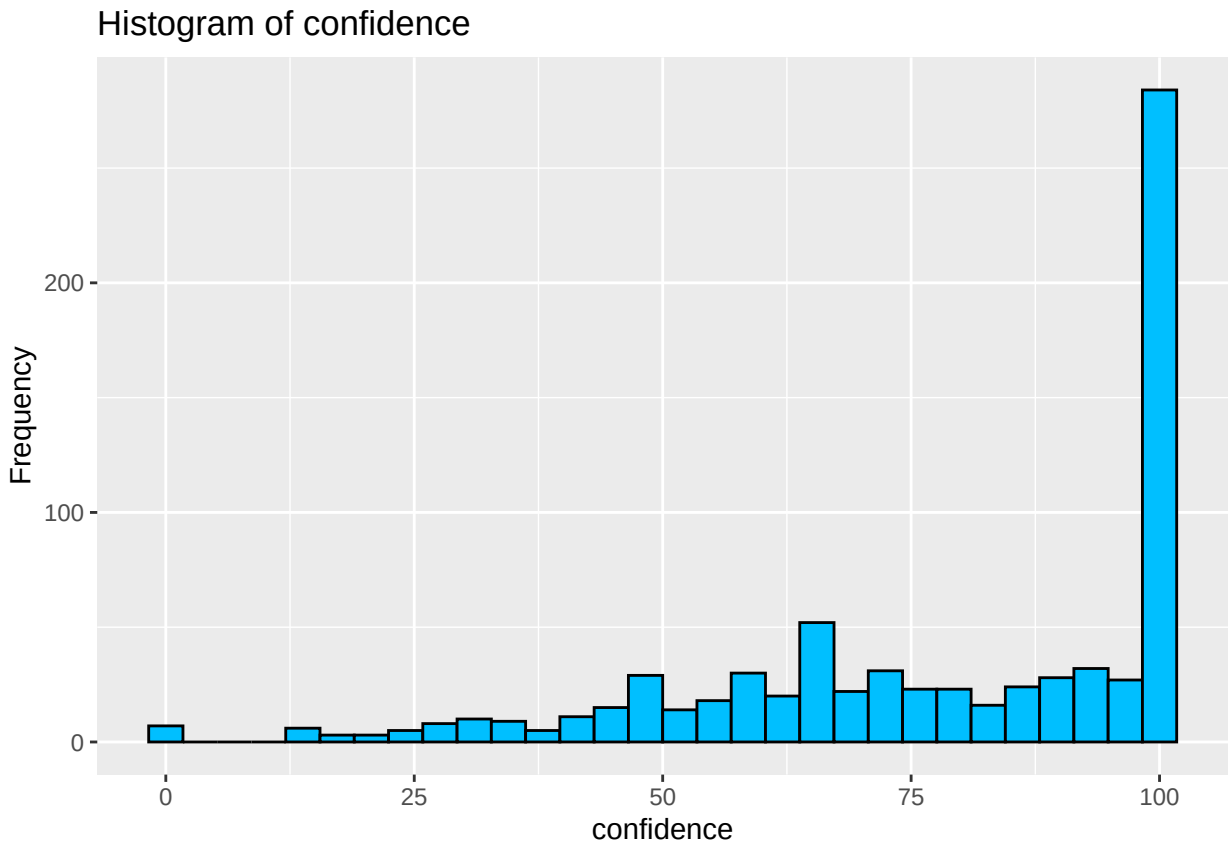

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



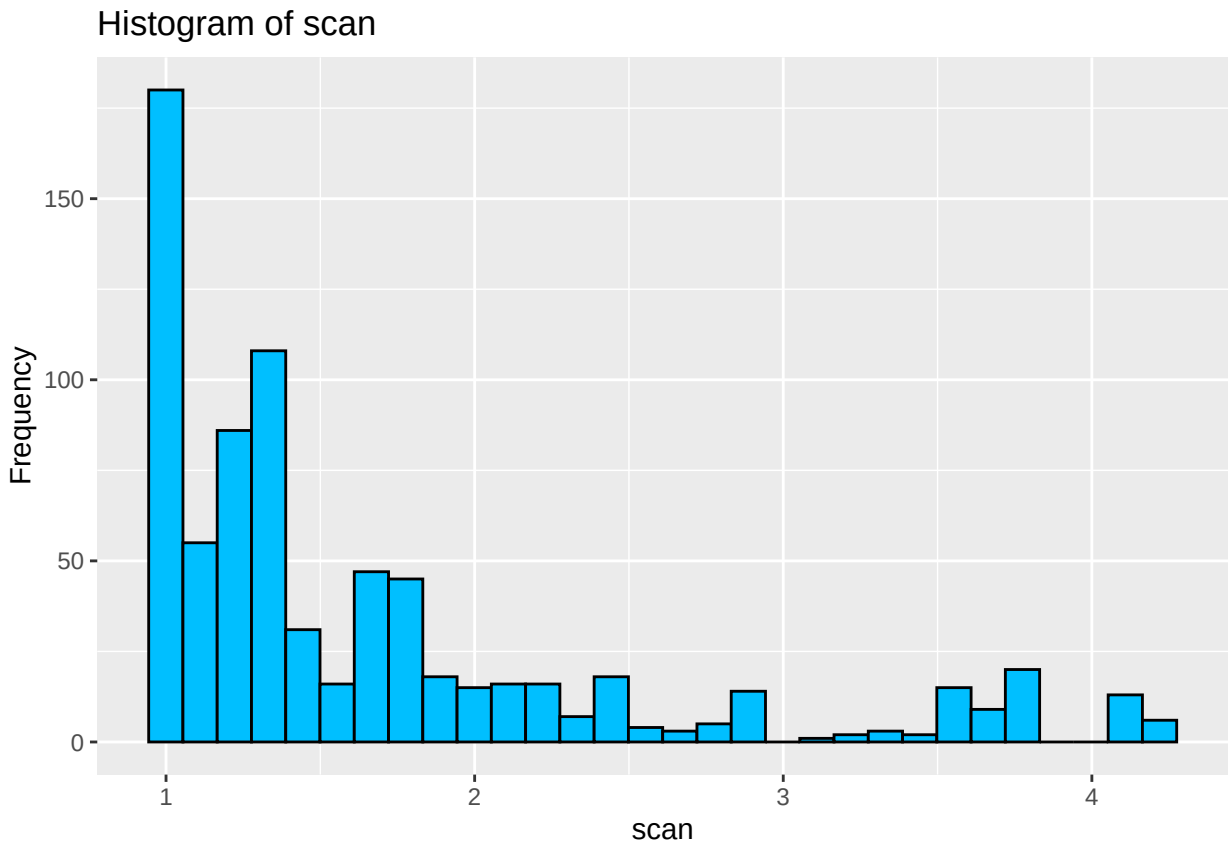

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



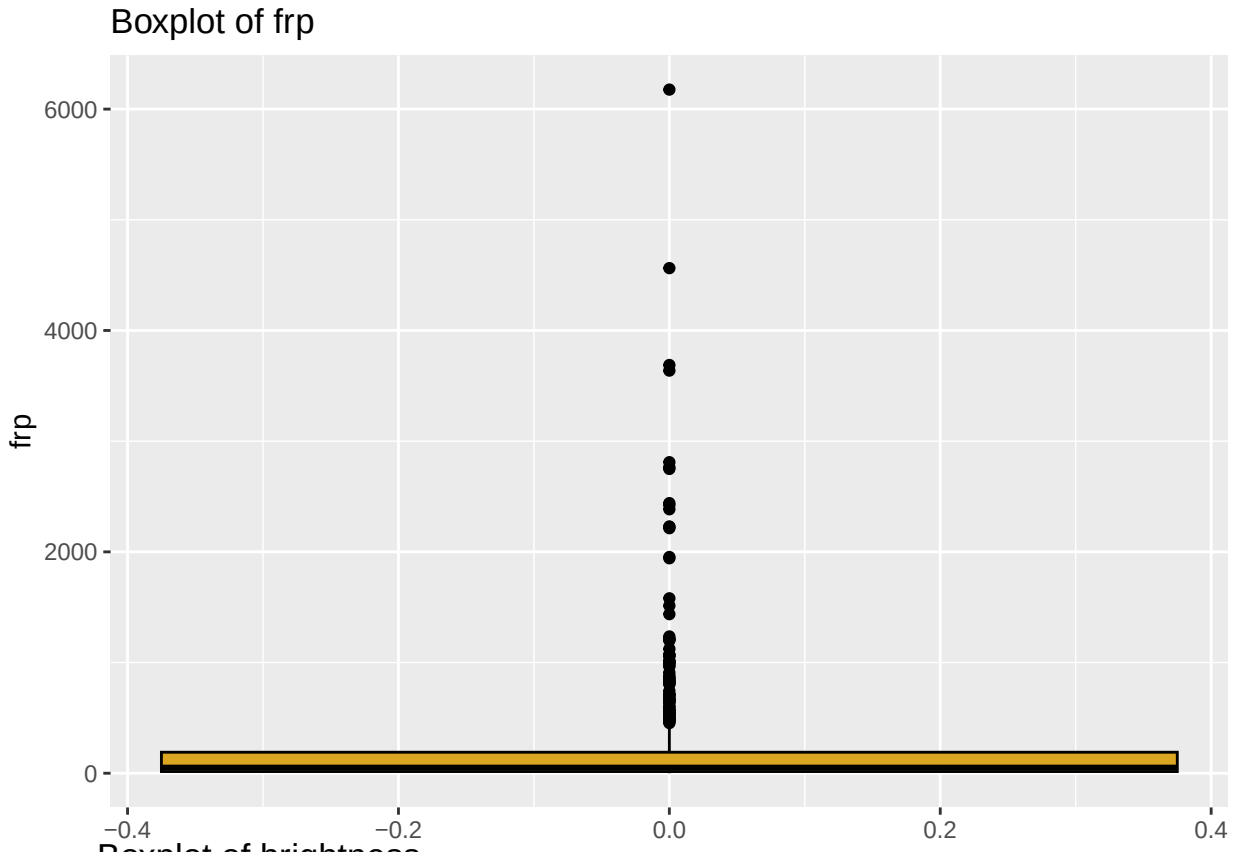

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

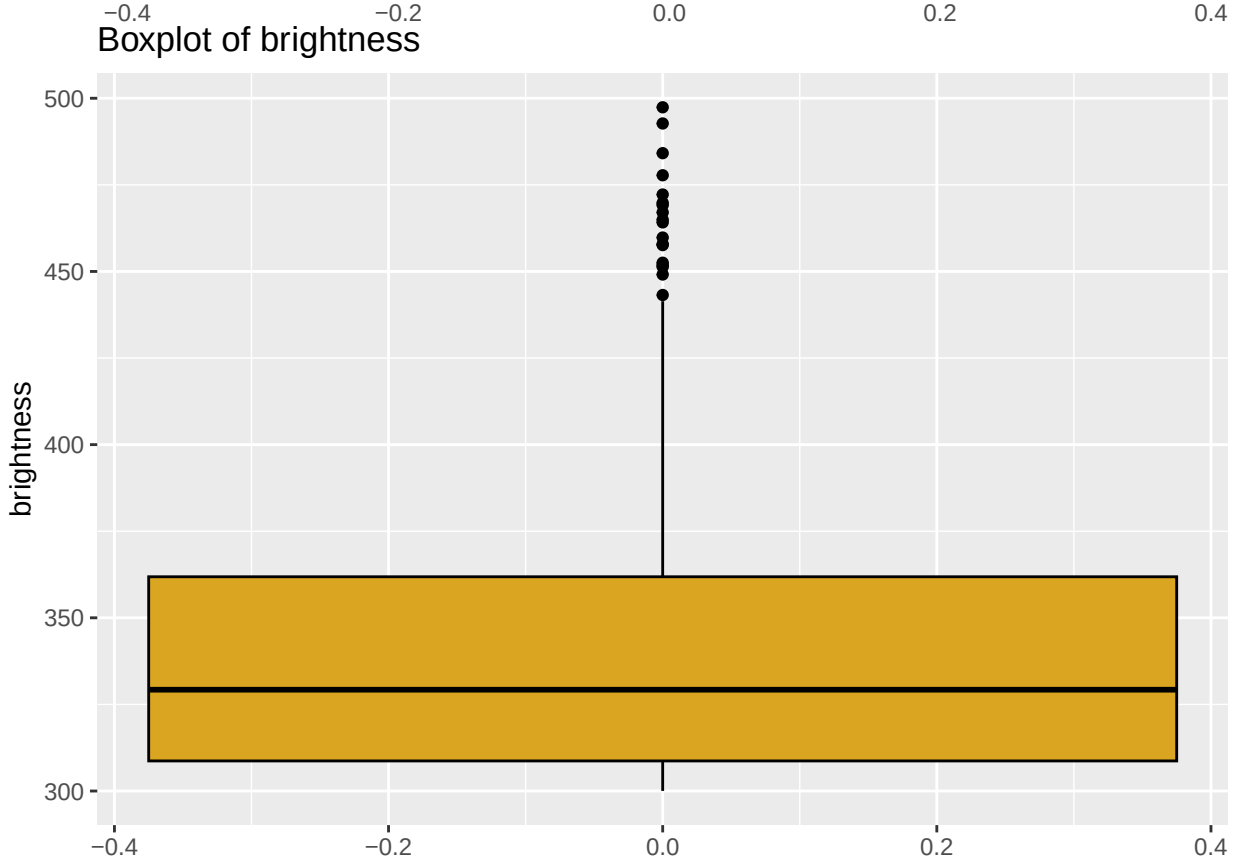


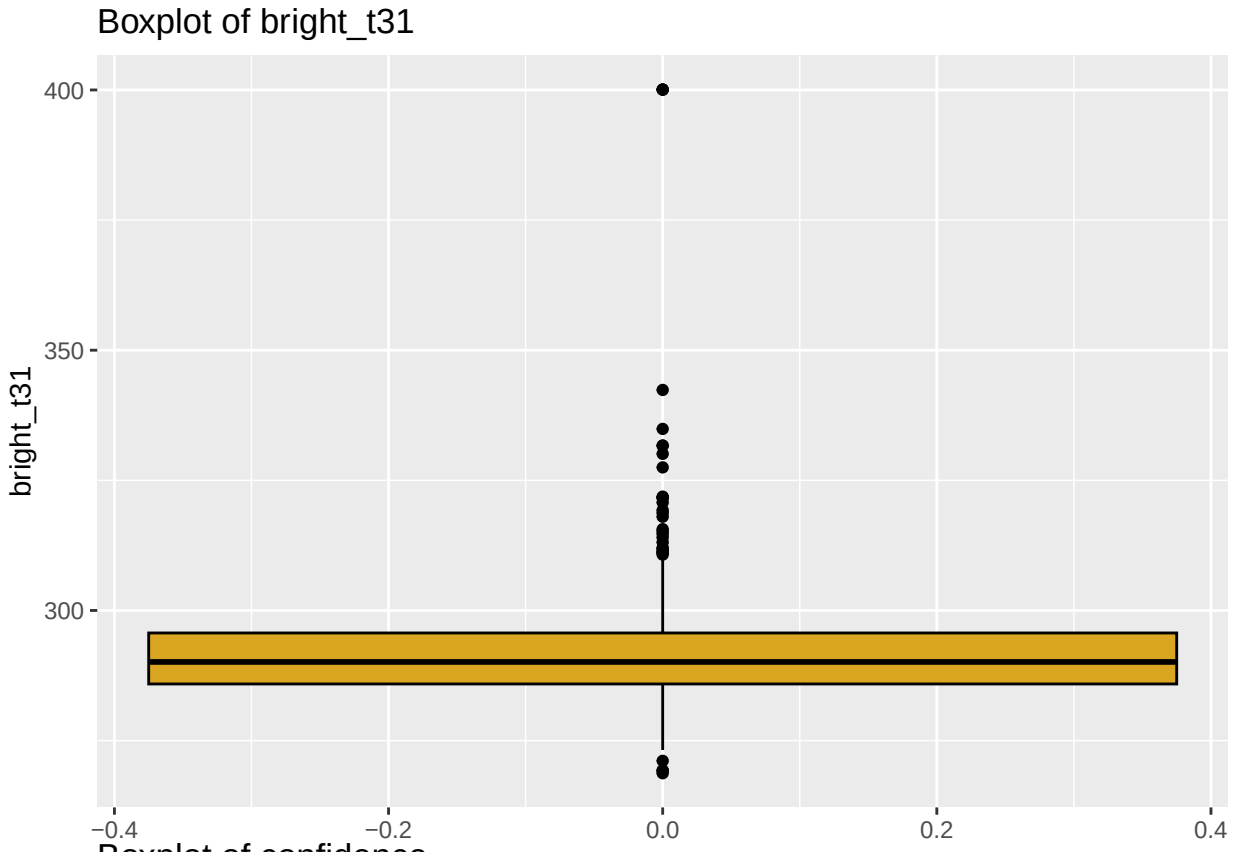

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

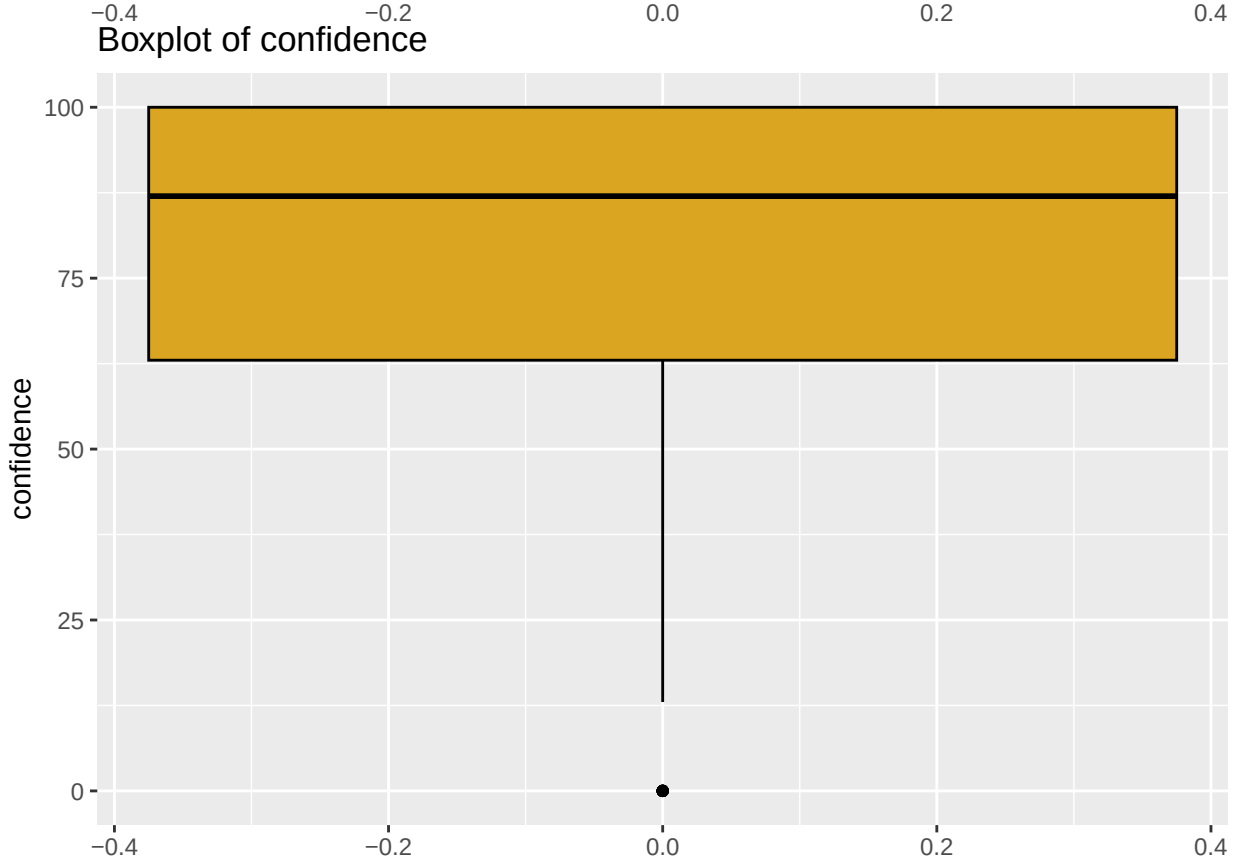


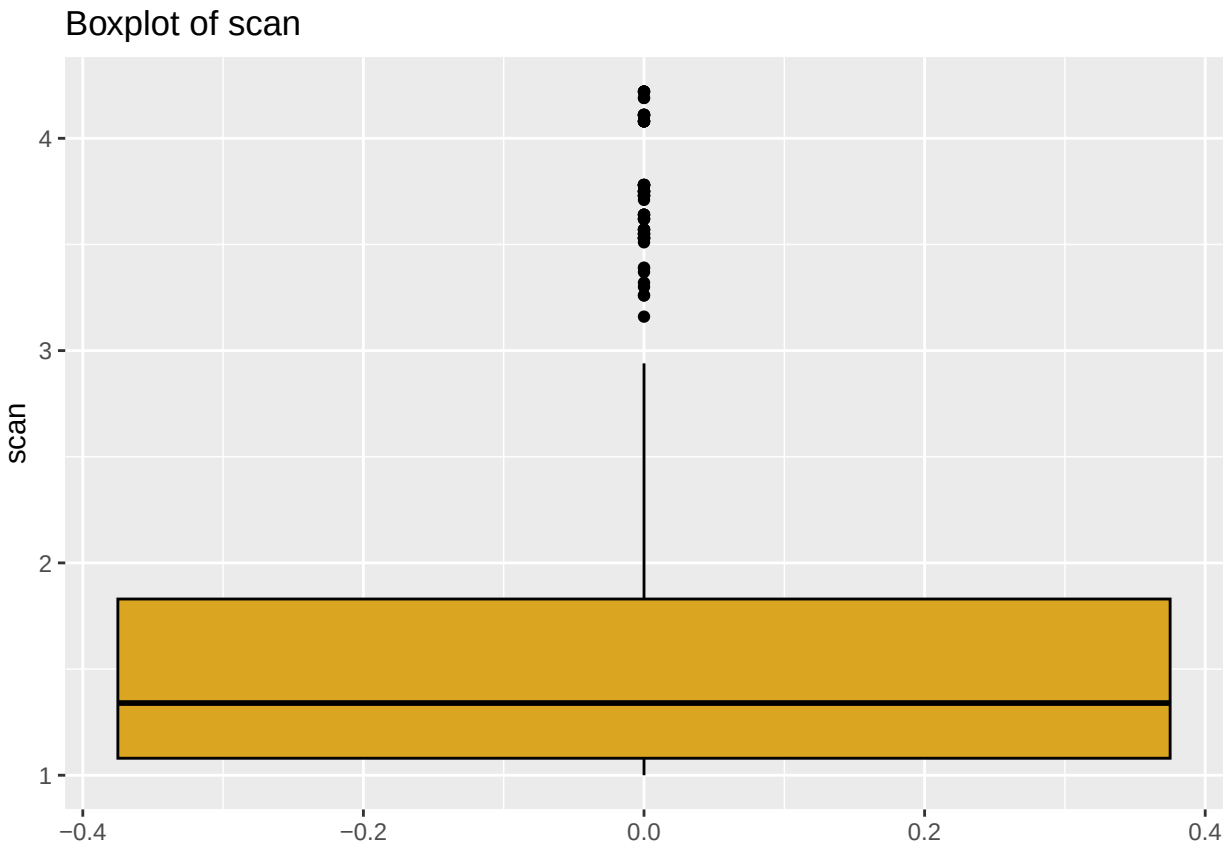

```
# Boxplots for numeric
for (var in numeric_vars) {
  p <- ggplot(final_data, aes_string(y = var)) +
    geom_boxplot(col = "black", fill = "goldenrod") +
    ggtitle(paste("Boxplot of", var)) +
    ylab(var)
  print(p)
}
```


Boxplot of frp
frp
6000
4000
2000
0
−0.4
−0.2
0.0
0.2
0.4

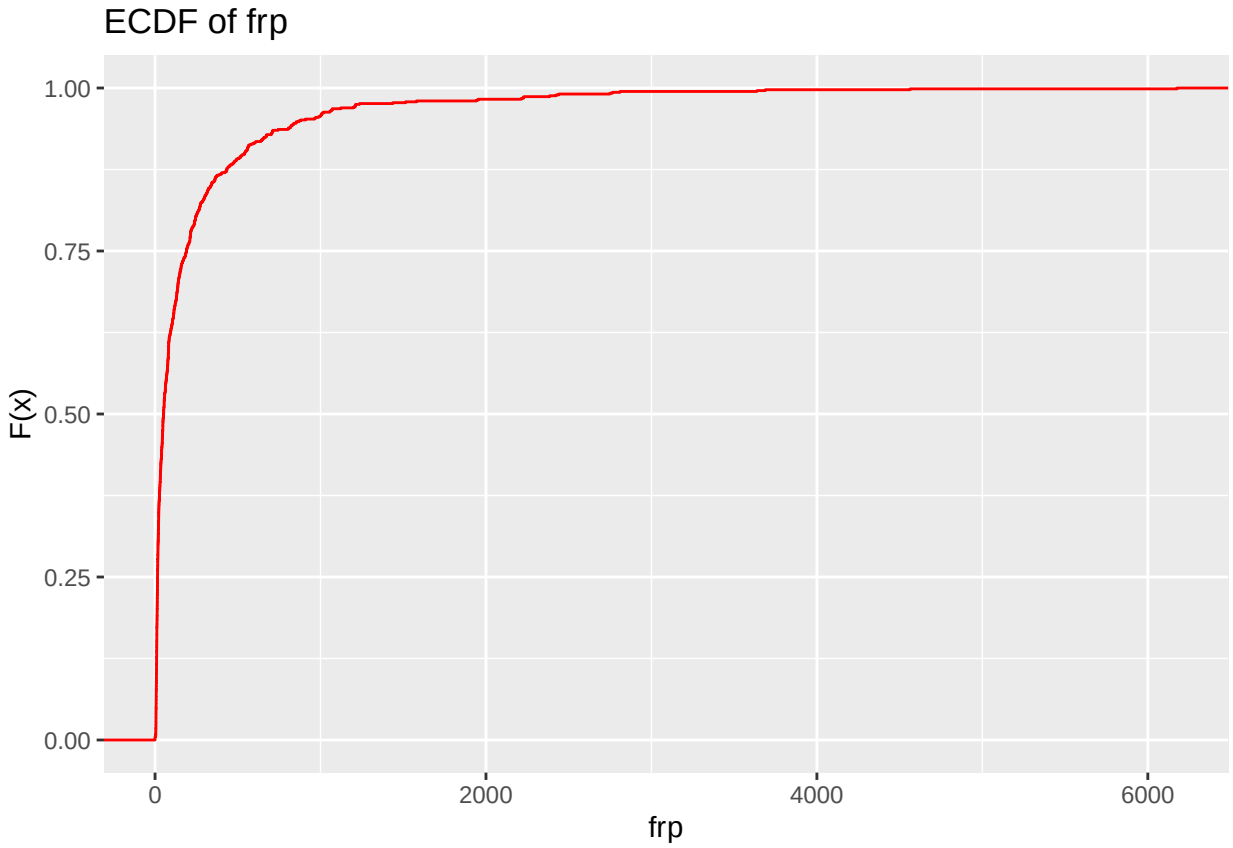

Boxplot of brightness
brightness
500
450
400
350
300
−0.4
−0.2
0.0
0.2
0.4

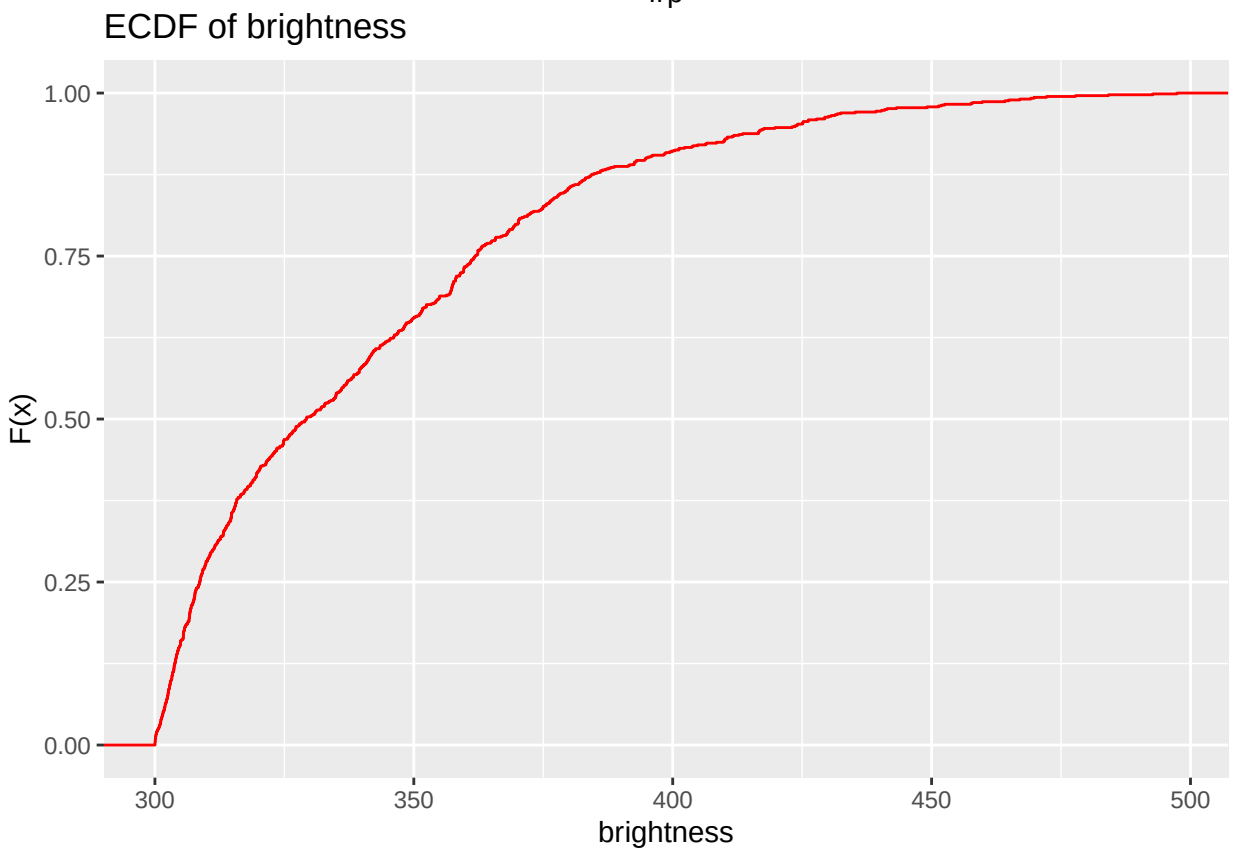

Boxplot of bright_t31
bright_t31
400
350
300
−0.4
−0.2
0.0
0.2
0.4

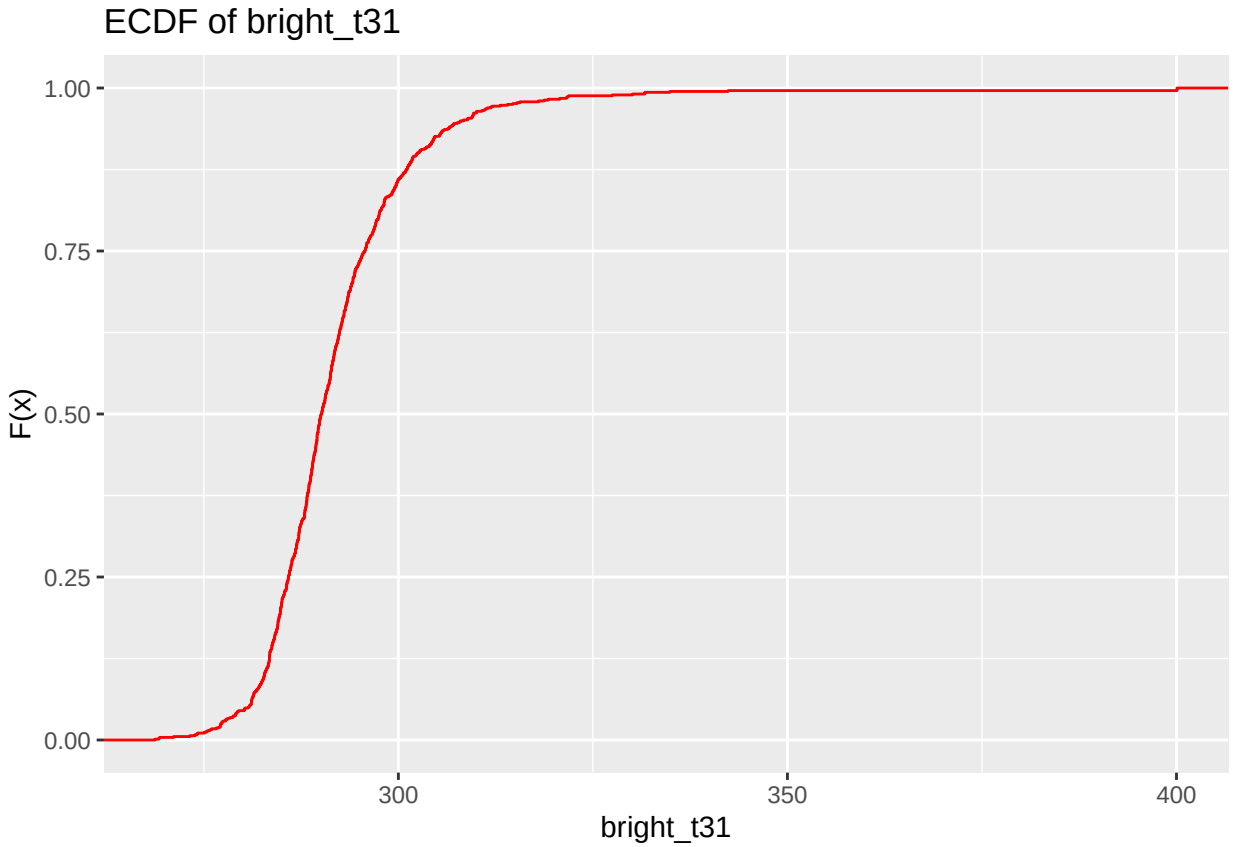

Boxplot of confidence
confidence
100
75
50
25
0
−0.4
−0.2
0.0
0.2
0.4

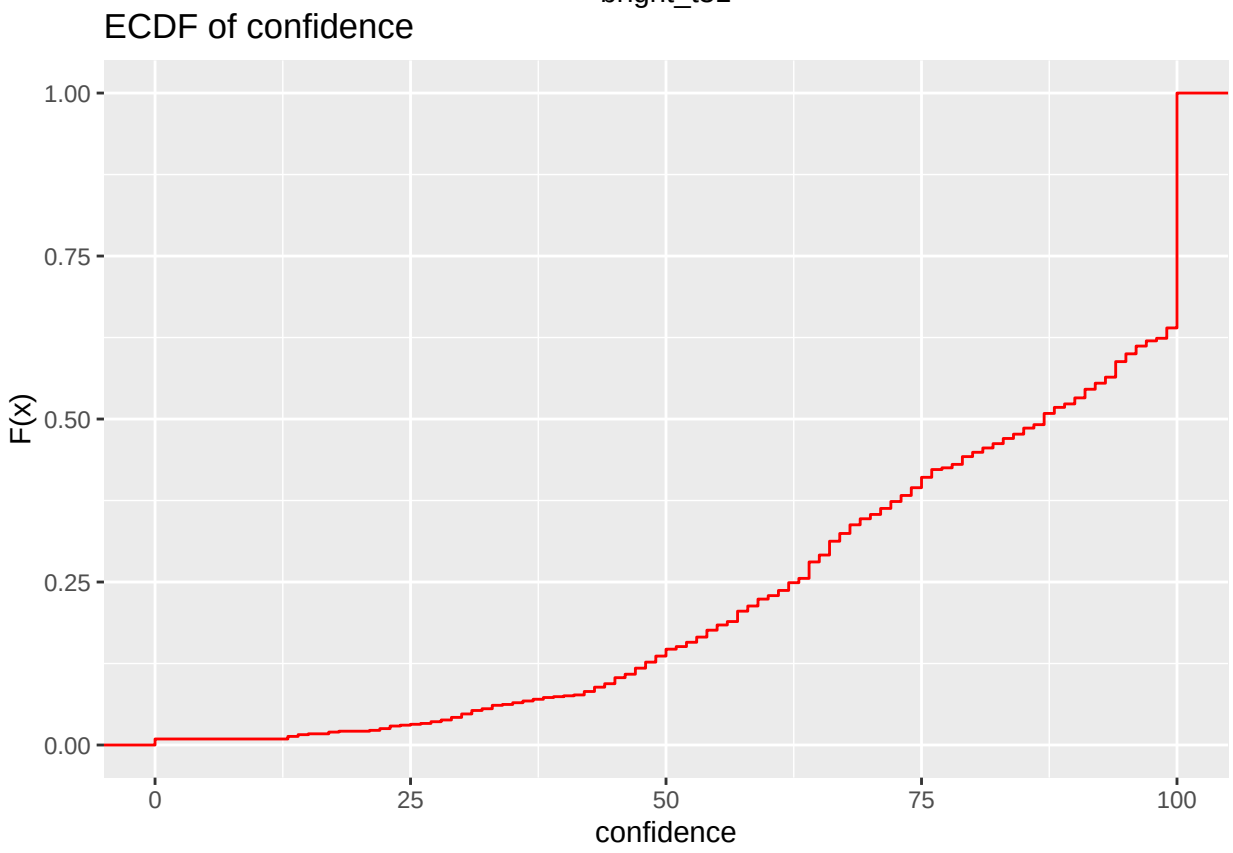


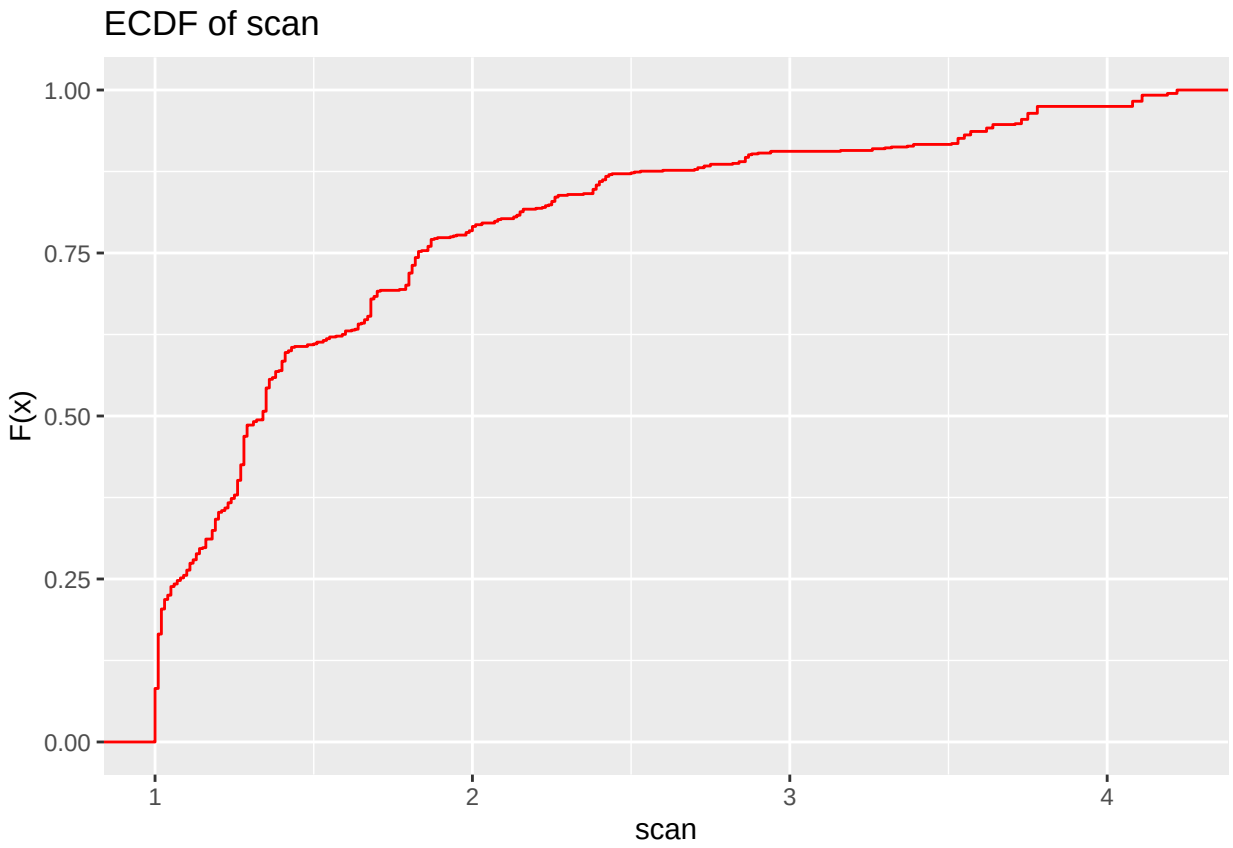

```r
# ECDFs
for (var in numeric_vars) {
  p <- ggplot(final_data, aes_string(x = var)) +
    stat_ecdf(col = "red") +
    ggtitle(paste("ECDF of", var)) +
    xlab(var) +
    ylab("F(x)")
  print(p)
}
```


ECDF of frp
1.00
0.75
0.50
0.25
0.00
F(x)
0
2000
4000
6000
frp

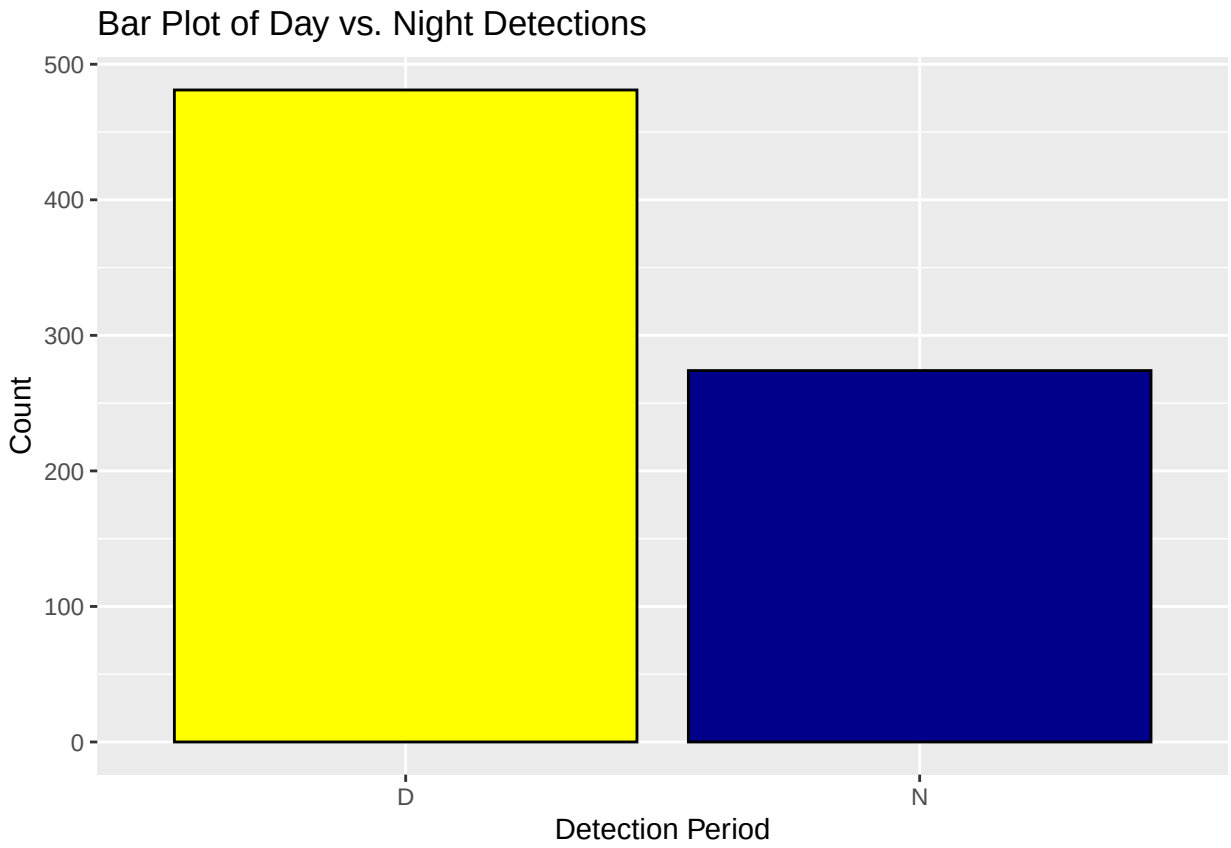

ECDF of brightness
1.00
0.75
0.50
0.25
0.00
F(x)
300
350
400
450
500
brightness


ECDF of bright_t31
1.00
0.75
0.50
0.25
0.00
F(x)
300
350
400
bright_t31

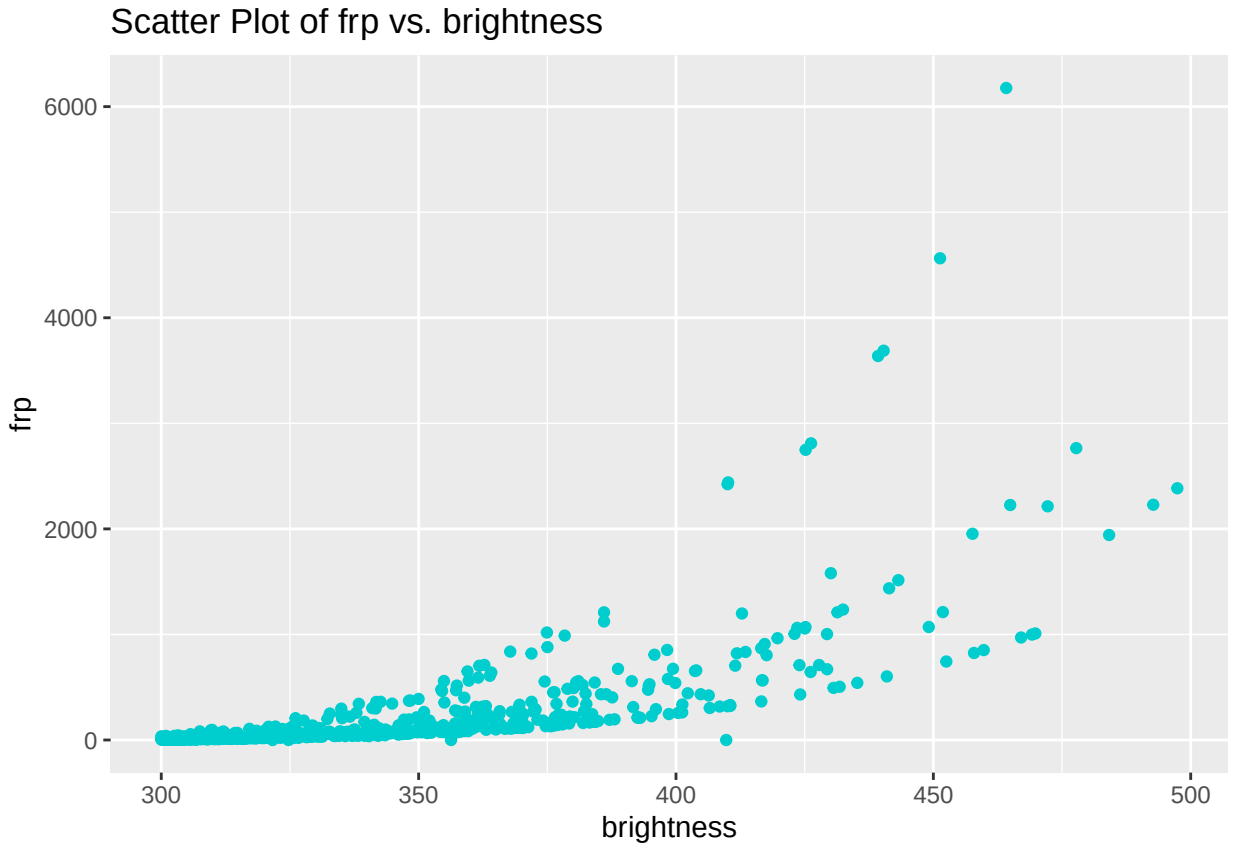

ECDF of confidence
1.00
0.75
0.50
0.25
0.00
F(x)
0
25
50
75
100
confidence

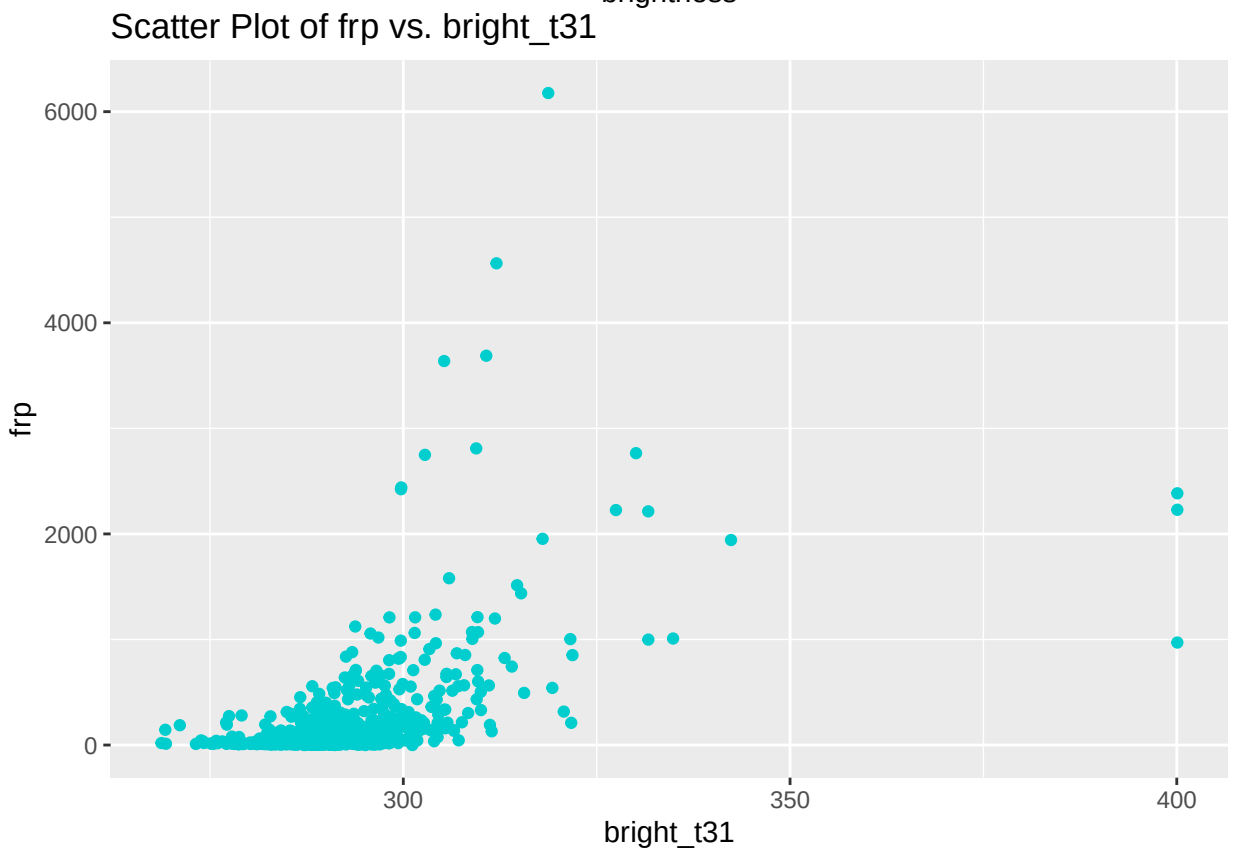


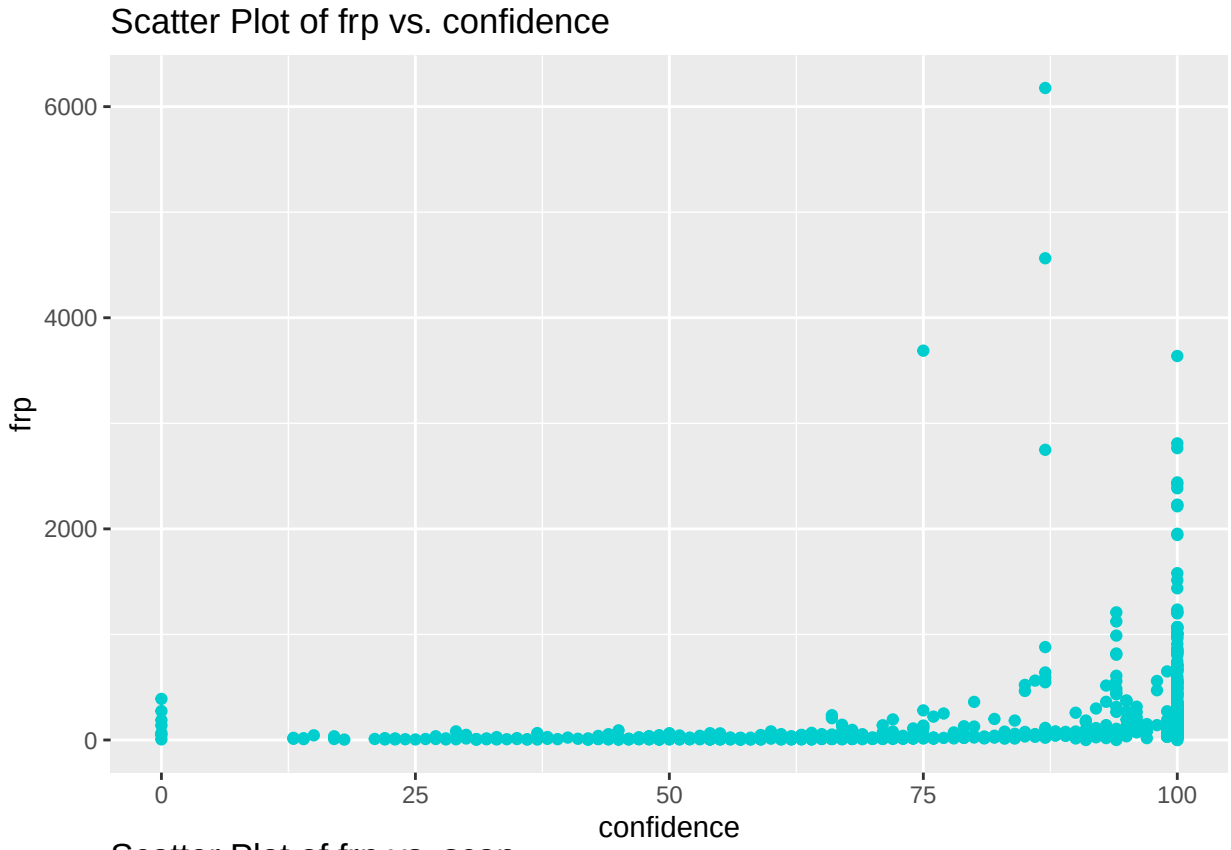

```
# Categoical barplot
ggplot(final_data, aes_string(x = cat_var)) +
  geom_bar(col = "black", fill = c("yellow", "darkblue")) +
  ggtitle("Bar Plot of Day vs. Night Detections") +
  xlab("Detection Period") +
  ylab("Count")
```

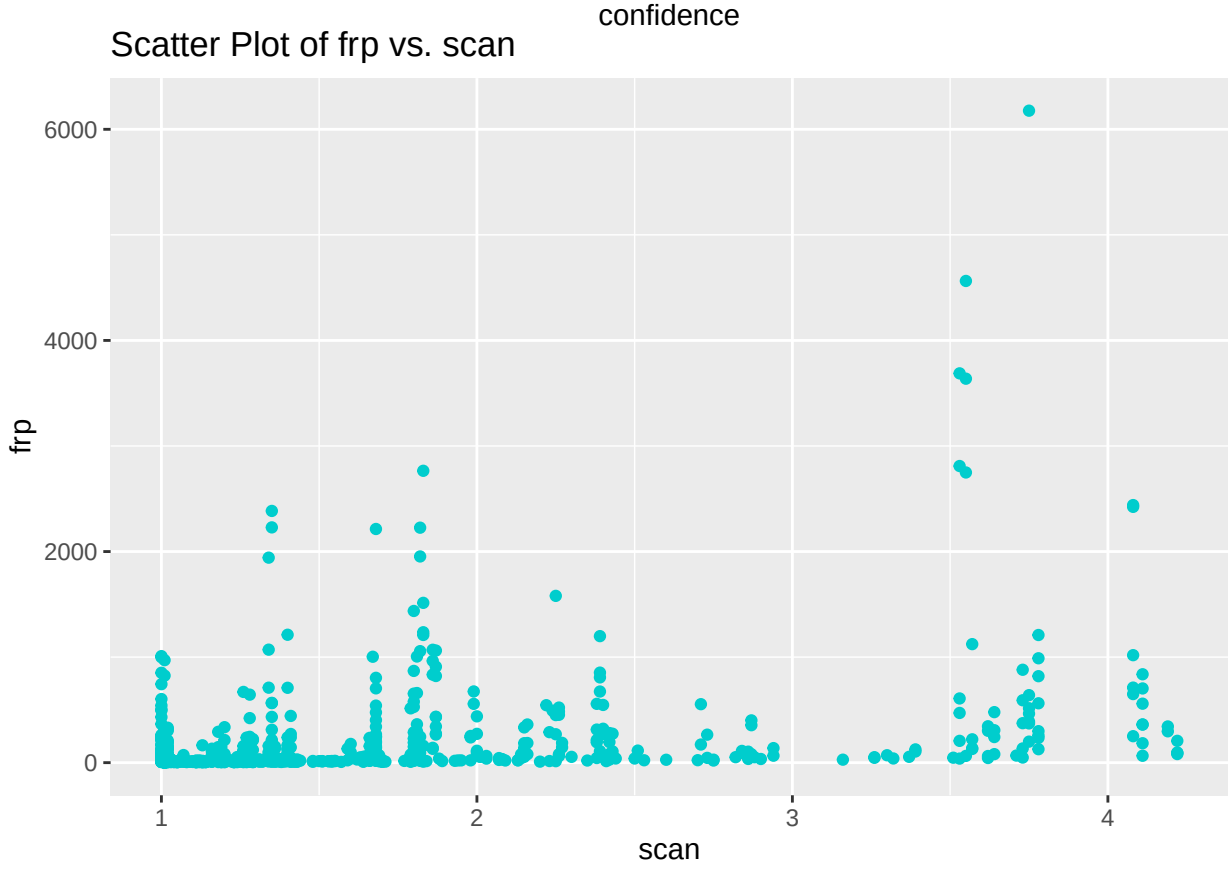



```
# frp vs. predictors
# Scatter Plots for FRP vs. numeric predictors
for (var in setdiff(numeric_vars, "frp")) {
  p <- ggplot(final_data, aes_string(x = var, y = "frp")) +
    geom_point(col = "cyan3") +
    ggtitle(paste("Scatter Plot of frp vs.", var)) +
    xlab(var) +
    ylab("frp")
  print(p)
}
```

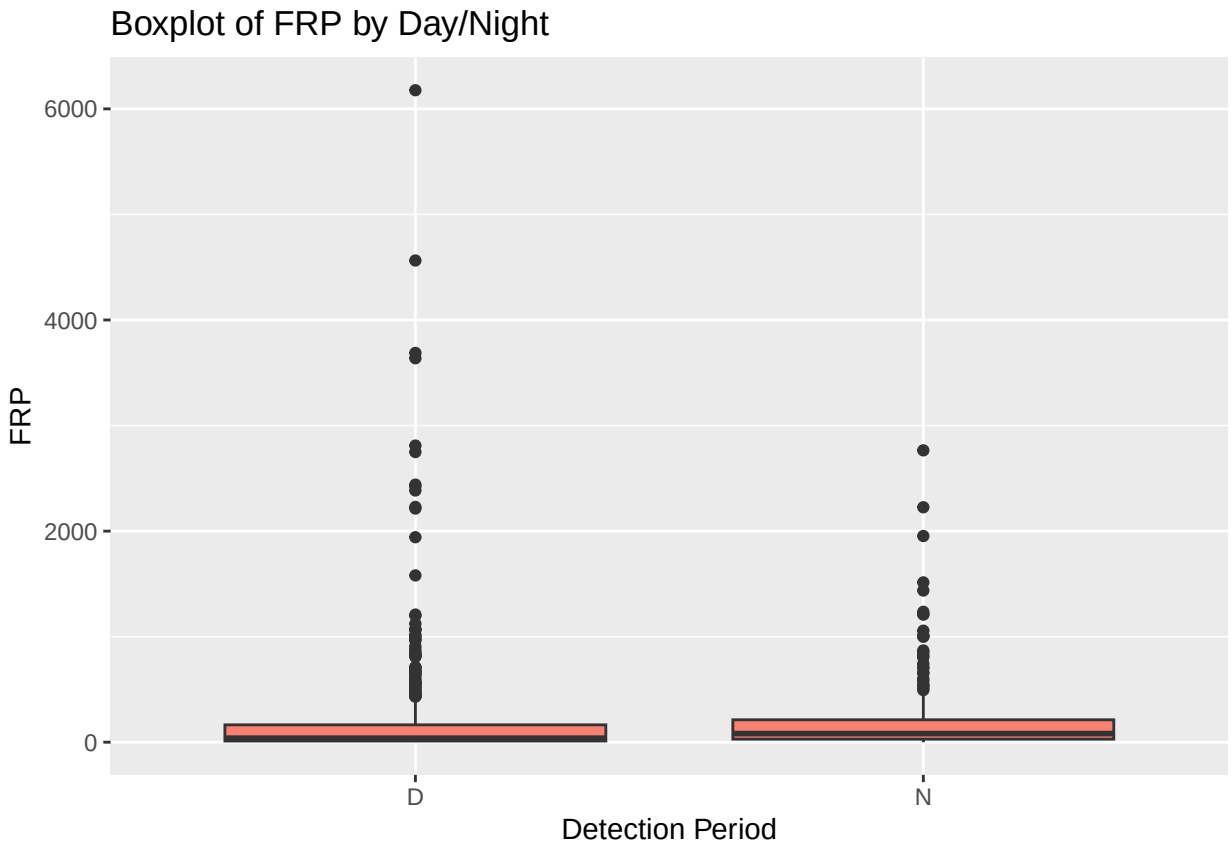

Scatter Plot of frp vs. brightness
6000
4000
2000
0
frp
300
350
400
450
500
brightness


Scatter Plot of frp vs. bright_t31
6000
4000
2000
0
frp
300
350
400
bright_t31


Scatter Plot of frp vs. confidence
6000
4000
2000
0
frp
0
25
50
75
100
confidence

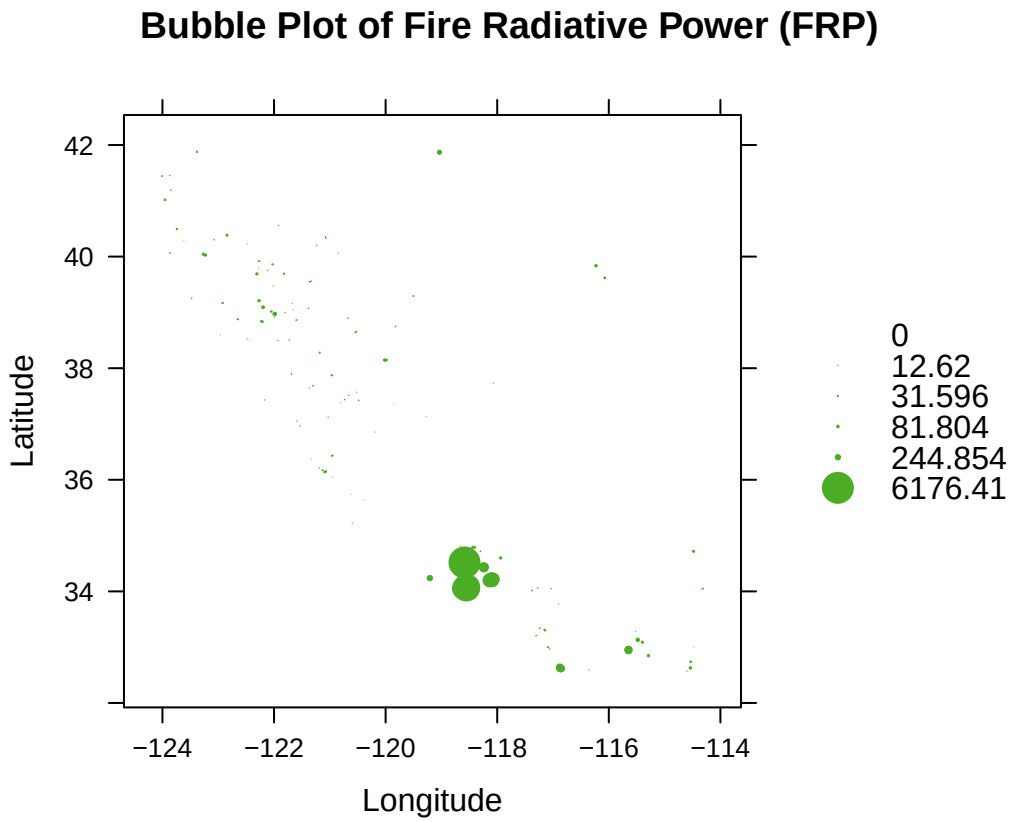

Scatter Plot of frp vs. scan
6000
4000
2000
0
frp
1
2
3
4
scan

```r
# Boxplot of FRP by "daynight"
ggplot(final_data, aes_string(x = cat_var, y = "frp")) +
  geom_boxplot(fill = "salmon") +
  ggtitle("Boxplot of FRP by Day/Night") +
  xlab("Detection Period") +
  ylab("FRP")
```

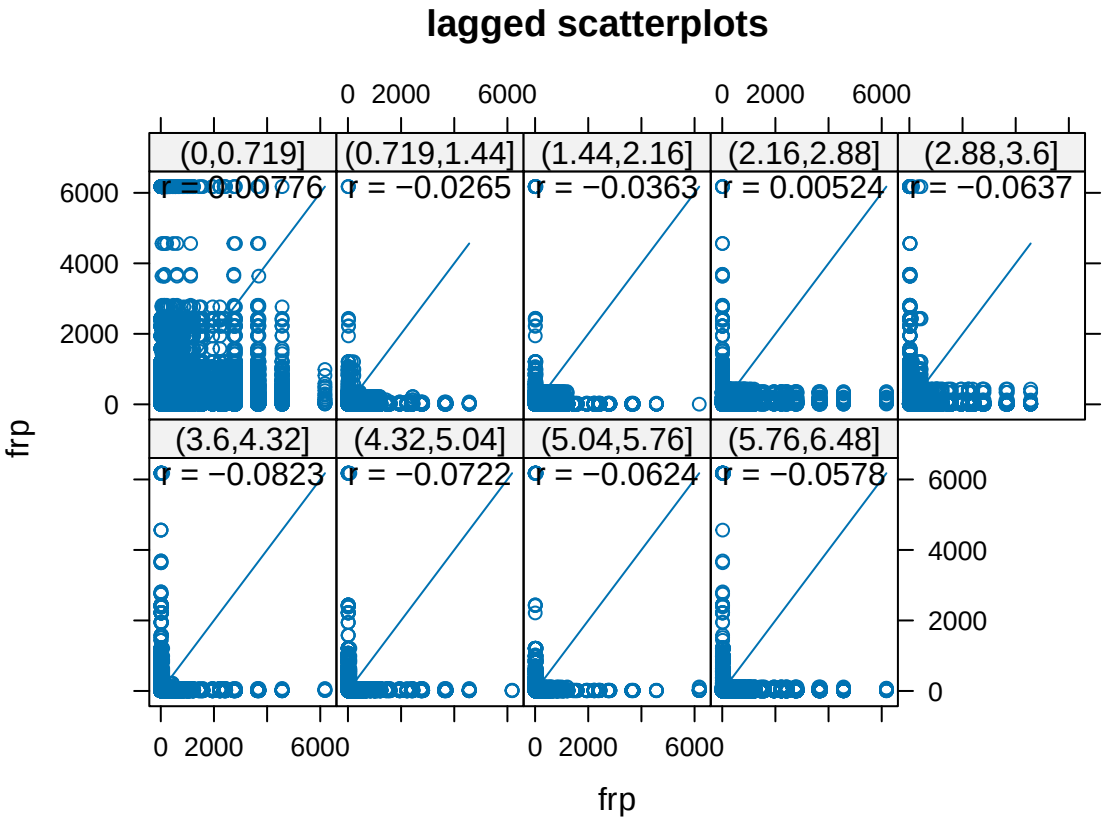


Bubble Plots and H-scatterplot

```
# Spatial data
final_data_sp <- final_data
coordinates(final_data_sp) <- ~ longitude + latitude

# Bubble plot for frp
bubble(final_data_sp, "frp",
       key.entries = quantile(final_data$frp, probs = seq(0, 1, 0.2), na.rm = TRUE),
       xlab = "Longitude",
       ylab = "Latitude",
       main = "Bubble Plot of Fire Radiative Power (FRP)",
       maxsize = 2,
       scales = list(draw = TRUE))
```




```
# H-scatterplot for frp
max_dist <- max(spDists(final_data_sp))
h_breaks <- seq(0, max_dist / 2, length.out = 10)
hscat(frp ~ 1, data = final_data_sp, breaks = h_breaks)
```


lagged scatterplots
(0,0.719]
(0.719,1.44]
(1.44,2.16]
(2.16,2.88]
(2.88,3.6]
(3.6,4.32]
(4.32,5.04]
(5.04,5.76]
(5.76,6.48]
r = 0.00776
r = −0.0265
r = −0.0363
r = 0.00524
r = −0.0637
r = −0.0823
r = −0.0722
r = −0.0624
r = −0.0578
0
2000
4000
6000
frp

Variograms

```
fire_geodata <- as.geodata(final_data,
                           coords.col = c("longitude", "latitude"),
                           data.col = "frp")
fire_max_dist <- max(dist(fire_geodata$coords))
fire_max_dist
```

```
## [1] 12.95001
```

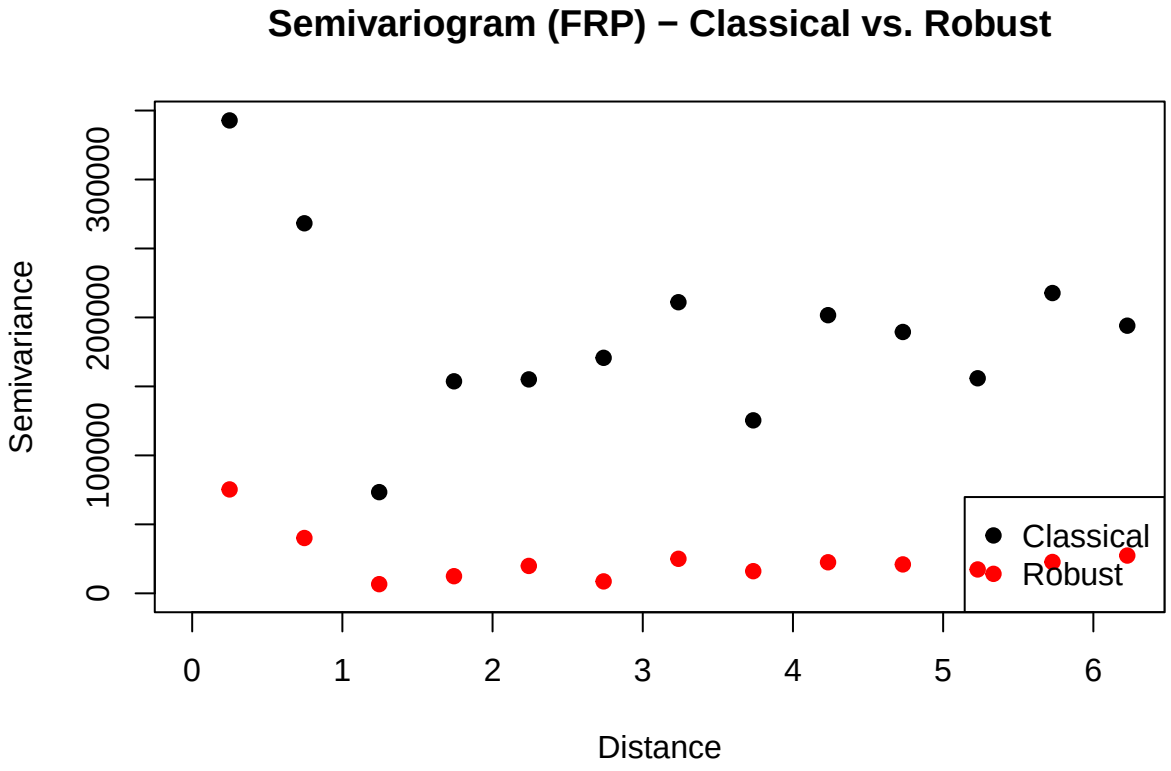
```
# Classical Semivariogram
fire_classical <- variog(fire_geodata,
                         max.dist = fire_max_dist / 2,
                         estimator.type = "classical")
```

```
## variog: computing omnidirectional variogram
```

```
# Robust  Semivariogram
fire_robust <- variog(fire_geodata,
                      max.dist = fire_max_dist / 2,
                      estimator.type = "modulus")
```

```
## variog: computing omnidirectional variogram
```

```
plot(fire_classical, pch = 19,
     main = "Semivariogram (FRP) - Classical vs. Robust",
     xlab = "Distance", ylab = "Semivariance")
points(fire_robust$u, fire_robust$v, pch = 19, col = "red")
legend("bottomright",
       legend = c("Classical", "Robust"),
       pch = 19,
       col = c("black", "red"))
```



```r
# --- Classical Fits ---
# By-Eye sph
plot(fire_classical, pch = 19,
     main = "Classical Semivariogram (FRP) with Various Fits",
     xlab = "Distance", ylab = "Semivariance")
lines.variomodel(cov.model = "sph",
                 cov.pars = c(150000, 3),
                 nugget = 50000,
                 max.dist = fire_max_dist / 2,
                 col = "red", lty = 2)

# Default weights sph
fire_fit_default <- variofit(fire_classical,
                             cov.model = "sph",
                             ini.cov.pars = c(150000, 4),
                             nugget = 50000)
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```r
fire_fit_default
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##     tausq   sigmasq       phi
##  97833.98 154731.75      0.00
## Practical Range with cor=0.05 for asymptotic range: 0
##
## variofit: minimised weighted sum of squares = 1.620933e+15
```

```r
lines(fire_fit_default, col = "blue", lty = 1)

# Cressie's weights
fire_fit_cressie <- variofit(fire_classical,
                             cov.model = "sph",
                             ini.cov.pars = c(150000, 3),
                             nugget = 50000,
                             weights = "cressie")
```

```
## variofit: covariance model used is spherical
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```r
fire_fit_cressie
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##    tausq  sigmasq      phi
## 120286.3 159463.6      0.0
## Practical Range with cor=0.05 for asymptotic range: 0
##
## variofit: minimised weighted sum of squares = 22941.44
```

```r
lines(fire_fit_cressie, col = "lightgreen", lty = 2)

# Equal Weights
fire_fit_equal <- variofit(fire_classical,
                           cov.model = "sph",
                           ini.cov.pars = c(150000, 3),
                           nugget = 50000,
                           weights = "equal")
```

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

```r
fire_fit_equal
```

```
## variofit: model parameters estimated by OLS (ordinary least squares):
## covariance model is: spherical
## parameter estimates:
##      tausq    sigmasq        phi
## 189163.097      0.000   4994.989
## Practical Range with cor=0.05 for asymptotic range: 4994.989
##
## variofit: minimised sum of squares = 52679573753
```

```r
lines(fire_fit_equal, col = "orange", lty = 2)

# MML
fire_fit_mml <- likfit(fire_geodata,
                       cov.model = "sph",
                       ini.cov.pars = c(150000, 3),
                       nugget = 50000,
                       lik.method = "ML")
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
fire_fit_mml
```

```
## likfit: estimated model parameters:
##        beta       tausq     sigmasq         phi
## "6.968e+01" "2.169e+05" "1.061e+04" "1.635e+00"
## Practical Range with cor=0.05 for asymptotic range: 1.635184
##
## likfit: maximised log-likelihood = -5718
```

```r
lines(fire_fit_mml, col = "purple", lty = 2)

# Exponential model
```

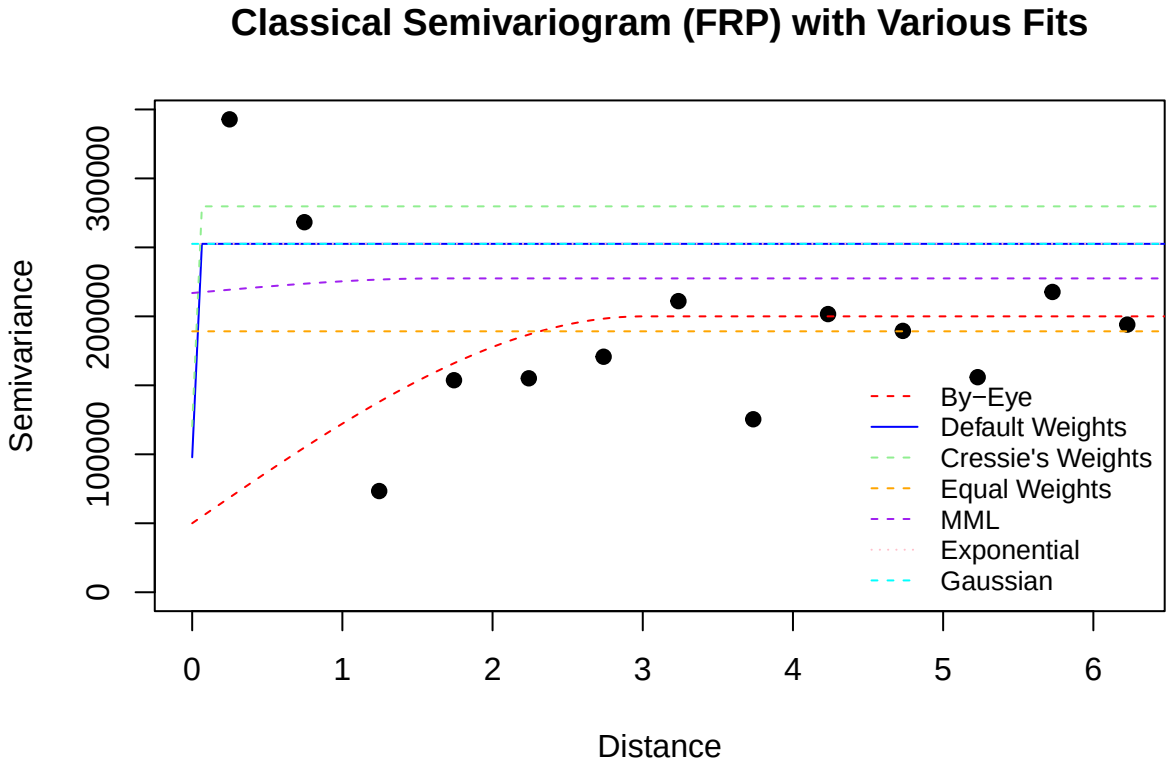
```
fire_fit_exp <- variofit(fire_classical,
                         cov.model = "exp",
                         ini.cov.pars = c(150000, 3),
                        nugget = 50000)
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(fire_fit_exp, col = "pink", lty = 3)

# Gaussian model
fire_fit_gaus <- variofit(fire_classical,
                          cov.model = "gau",
                          ini.cov.pars = c(150000, 3),
                          nugget = 50000)
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(fire_fit_gaus, col = "cyan", lty = 2)

legend("bottomright",
       legend = c("By-Eye",
                  "Default Weights",
                  "Cressie's Weights",
                  "Equal Weights",
                  "MML",
                  "Exponential",
                  "Gaussian"),
       col = c("red", "blue", "lightgreen", "orange", "purple", "pink", "cyan"),
       lty = c(2, 1, 2, 2, 2, 3, 2),
       bty = "n",
       cex = 0.8)
```




```
# --- Robust Fits ---
plot(fire_robust, pch = 19,
     main = "Robust Semivariogram (FRP) with Various Fits",
     xlab = "Distance", ylab = "Semivariance")

# By-eye
lines.variomodel(cov.model = "sph",
                 cov.pars = c(20000, 3),
                 nug = 0,
                 max.dist = fire_max_dist / 2,
                 col = "red", lty = 2)

# Default weights
fire_fit_default_rb <- variofit(fire_robust,
                                cov.model = "sph",
                                ini.cov.pars = c(20000, 3),
                                nugget = 0)
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(fire_robust, cov.model = "sph", ini.cov.pars = c(20000, :
## unreasonable initial value for sigmasq + nugget (too low)
```

```
fire_fit_default_rb
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##    tausq  sigmasq      phi
```

```
## 20809.94 23514.48     0.00
## Practical Range with cor=0.05 for asymptotic range: 0
##
## variofit: minimised weighted sum of squares = 1.625135e+14
```

```
lines(fire_fit_default_rb, col = "blue", lty = 1)

# Cressie's weights
fire_fit_cressie_rb <- variofit(fire_robust,
                            cov.model = "sph",
                            ini.cov.pars = c(20000, 3),
                            nugget = 0,
                            weights = "cressie")
```

```
## variofit: covariance model used is spherical
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
## Warning in variofit(fire_robust, cov.model = "sph", ini.cov.pars = c(20000, :
## unreasonable initial value for sigmasq + nugget (too low)
```

```
fire_fit_cressie_rb
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##    tausq  sigmasq      phi
## 35427.99 24426.47     0.00
## Practical Range with cor=0.05 for asymptotic range: 0
##
## variofit: minimised weighted sum of squares = 61256.16
```

```
lines(fire_fit_cressie_rb, col = "lightgreen", lty = 2)

# Equal Weights
fire_fit_equal_rb <- variofit(fire_robust,
                            cov.model = "sph",
                            ini.cov.pars = c(20000, 3),
                            nugget = 0,
                            weights = "equal")
```

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

```
## Warning in variofit(fire_robust, cov.model = "sph", ini.cov.pars = c(20000, :
## unreasonable initial value for sigmasq + nugget (too low)
```

```
fire_fit_equal_rb
```

```
## variofit: model parameters estimated by OLS (ordinary least squares):
## covariance model is: spherical
## parameter estimates:
##     tausq   sigmasq       phi
##  3581.682 20651.000     0.000
## Practical Range with cor=0.05 for asymptotic range: 0
##
## variofit: minimised sum of squares = 3714515703
```

```
lines(fire_fit_equal_rb, col = "orange", lty = 2)

# MML
fire_fit_mml_rb <- likfit(fire_geodata,
                          cov.model = "sph",
                          ini.cov.pars = c(20000, 3),
                          nugget = 0,
                          lik.method = "ML")
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```
fire_fit_mml_rb
```

```
## likfit: estimated model parameters:
##        beta       tausq     sigmasq         phi
## "6.968e+01" "2.169e+05" "1.061e+04" "1.635e+00"
## Practical Range with cor=0.05 for asymptotic range: 1.634856
##
## likfit: maximised log-likelihood = -5718
```

```
lines(fire_fit_mml_rb, col = "purple", lty = 2)

# Exponential model
fire_fit_exp_rb <- variofit(fire_robust,
                            cov.model = "exp",
                            ini.cov.pars = c(20000, 3),
                            nugget = 0)
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

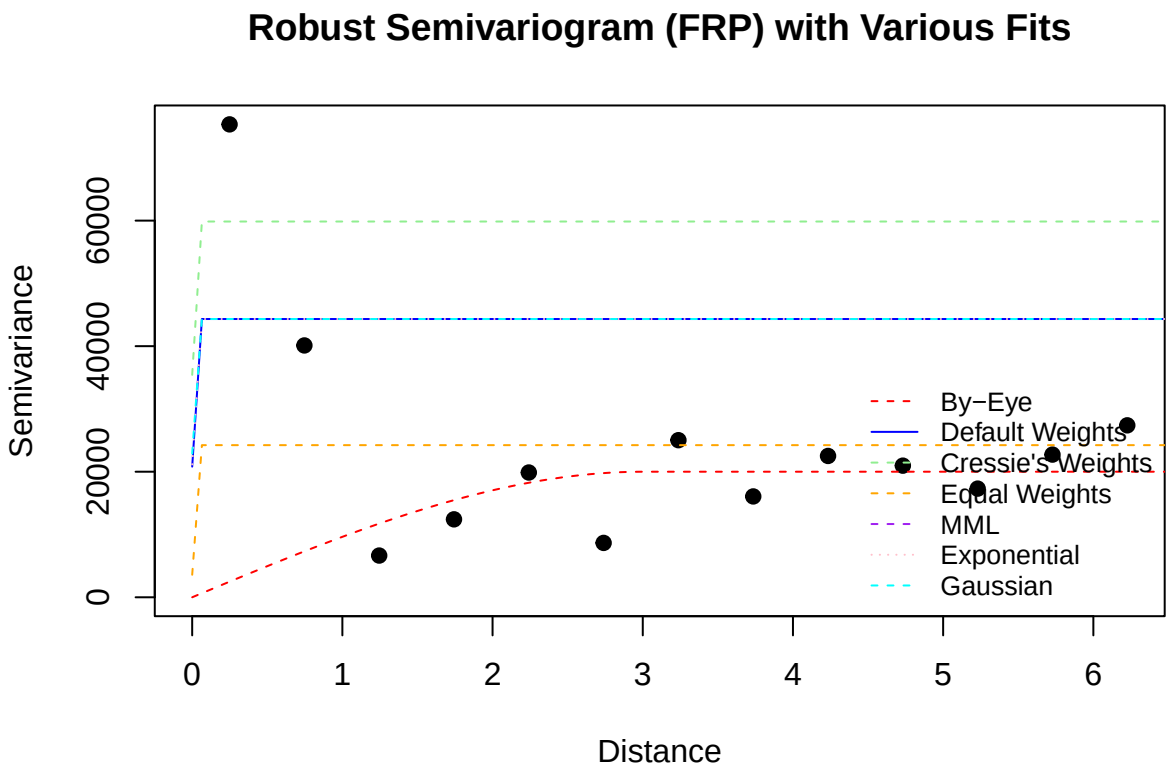
```
## Warning in variofit(fire_robust, cov.model = "exp", ini.cov.pars = c(20000, :
## unreasonable initial value for sigmasq + nugget (too low)
```

```
lines(fire_fit_exp_rb, col = "pink", lty = 3)

# Gaussian model
fire_fit_gaus_rb <- variofit(fire_robust,
                             cov.model = "gau",
                             ini.cov.pars = c(20000, 3),
                             nugget = 0)
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(fire_robust, cov.model = "gau", ini.cov.pars = c(20000, :
## unreasonable initial value for sigmasq + nugget (too low)
```

```
lines(fire_fit_gaus_rb, col = "cyan", lty = 2)

legend("bottomright",
       legend = c("By-Eye",
                  "Default Weights",
                  "Cressie's Weights",
                  "Equal Weights",
                  "MML",
                  "Exponential",
                  "Gaussian"),
       col = c("red", "blue", "lightgreen", "orange", "purple", "pink", "cyan"),
       lty = c(2, 1, 2, 2, 2, 3, 2),
       bty = "n",
       cex = 0.8)
```

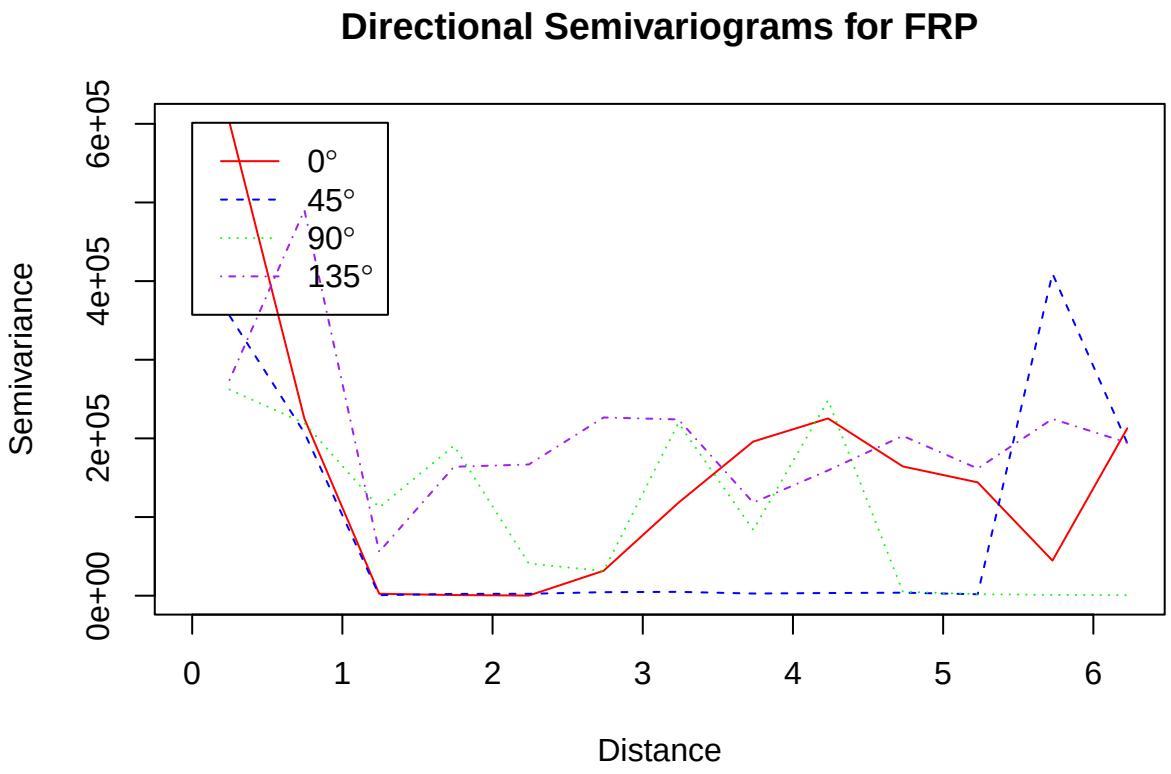



```
# --- Directional semivariogram ---
dir_vario <- variog4(fire_geodata,
                     max.dist = fire_max_dist / 2,
                     direction = c(0, 45, 90, 135),  # cardinal directions
                     tolerance = 22.5,
                     unit.angle = "degrees")
```

```
## variog: computing variogram for direction = 0 degrees (0 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 45 degrees (0.785 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 90 degrees (1.571 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
```

```
## variog: computing variogram for direction = 135 degrees (2.356 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing omnidirectional variogram
```

```r
plot(dir_vario,
     col = c("red", "blue", "green", "purple"),
     pch = 19,
     main = "Directional Semivariograms for FRP",
     xlab = "Distance",
     ylab = "Semivariance")
title(main = "Directional Semivariograms for FRP")
```

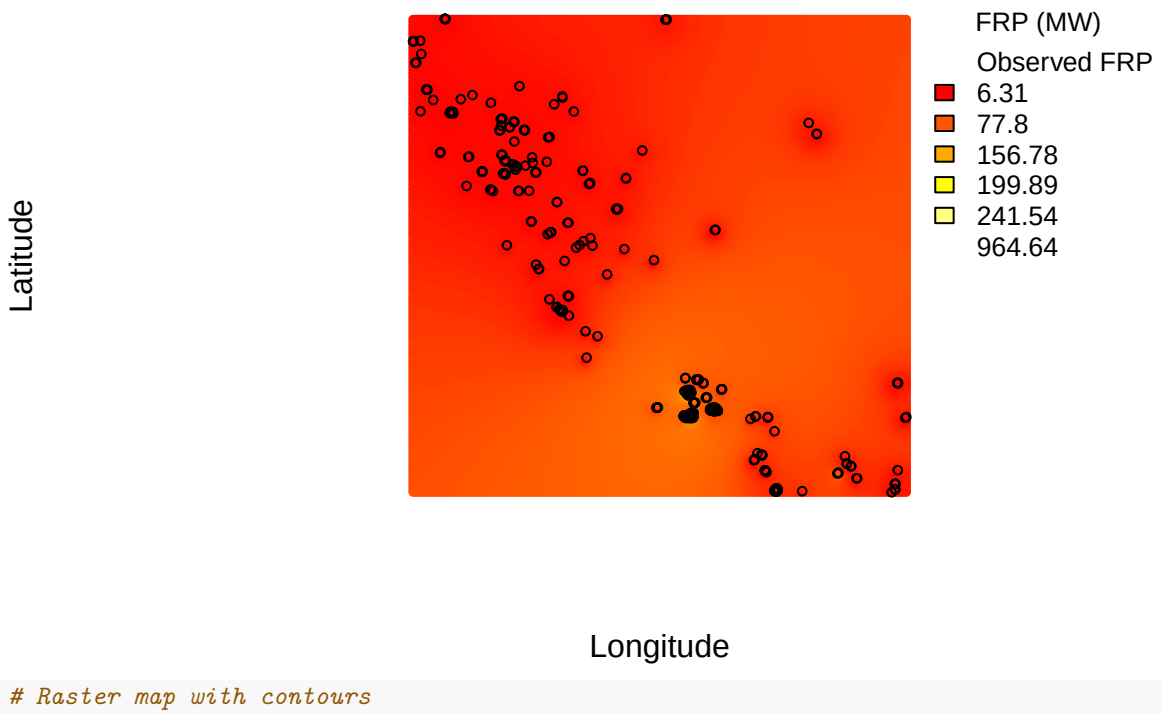


IDW Interpolation Prediction

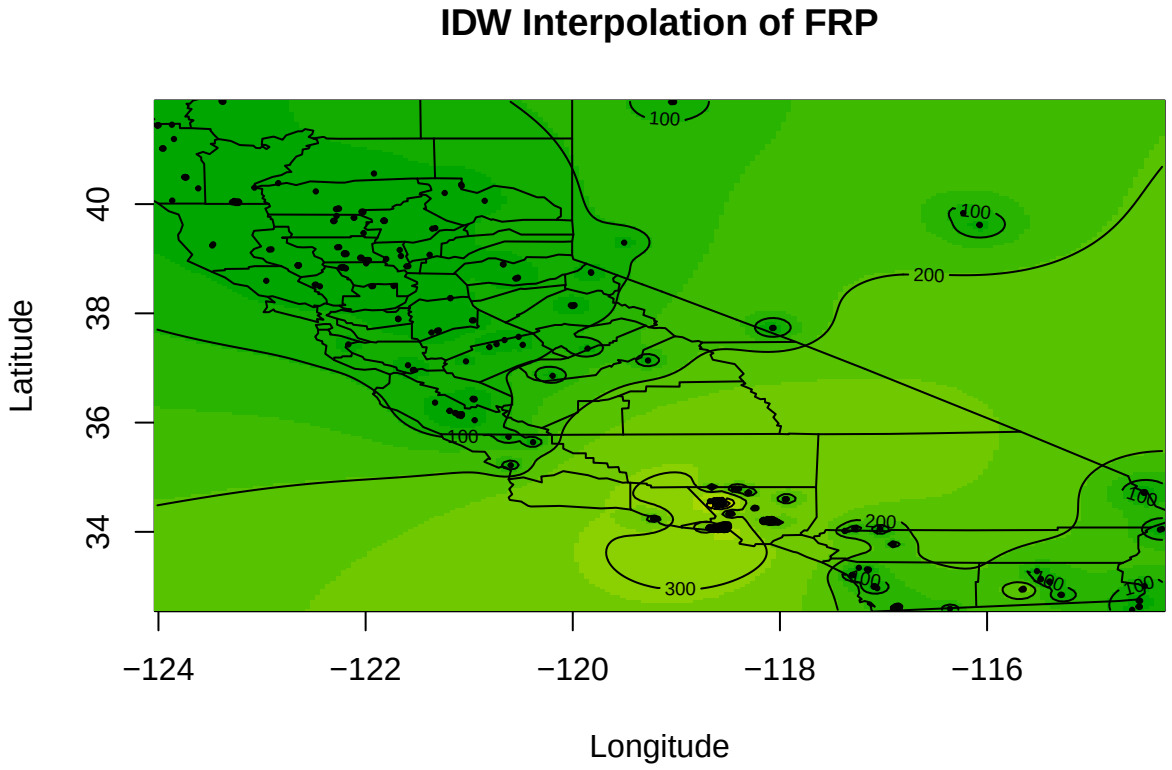
```
# Interpolation method predictions
x.range <- range(final_data_sp$longitude)
y.range <- range(final_data_sp$latitude)

dense_grid <- expand.grid(
  longitude = seq(from = x.range[1], to = x.range[2], by = 0.05),
  latitude = seq(from = y.range[1], to = y.range[2], by = 0.05)
)
coordinates(dense_grid) <- ~ longitude + latitude


# Predict data points - idw = inverse distance weighted predictor
idw_frp <- idw(frp ~ 1,
               locations = final_data_sp,
               newdata = dense_grid)
```

```
## [inverse distance weighted interpolation]
```

```
# --- Plot ---
# Scatter plot with predictions
plot(dense_grid, pch = ".", col = "gray",
     main = "IDW Predictions of FRP",
     xlab = "Longitude", ylab = "Latitude")
points(idw_frp, pch = 19, cex = 0.5,
       col = heat.colors(100)[cut(idw_frp$var1.pred, breaks = 100)])
points(final_data_sp, pch = 1, col = "black", cex = 0.6)
legend("topright",
       legend = c("Observed FRP", round(quantile(idw_frp$var1.pred, probs = seq(0, 1, 0.2)), 2)),
       fill = c(NA, heat.colors(5)),
       border = c(NA, rep("black", 5)),
       title = "FRP (MW)",
       bty = "n",
       cex = 0.8)
```

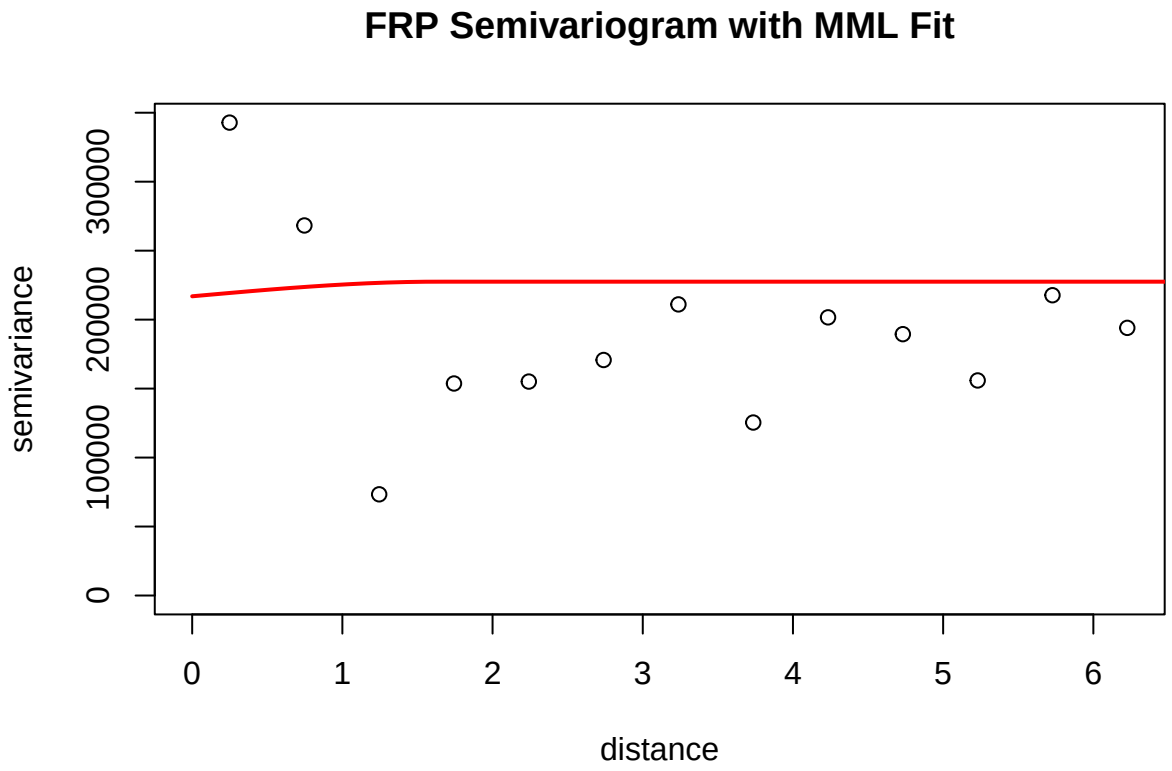



```
# Raster map with contours
idw_df <- as.data.frame(idw_frp)
idw_mat <- matrix(idw_df$var1.pred,
                  nrow = length(seq(from = x.range[1], to = x.range[2], by = 0.05)),
                  ncol = length(seq(from = y.range[1], to = y.range[2], by = 0.05)))

image(x = seq(x.range[1], x.range[2], by = 0.05),
      y = seq(y.range[1], y.range[2], by = 0.05),
      z = idw_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "IDW Interpolation of FRP")
contour(x = seq(x.range[1], x.range[2], by = 0.05),
        y = seq(y.range[1], y.range[2], by = 0.05),
        z = idw_mat,
        add = TRUE,
        nlevels = 10)
points(final_data_sp, pch = 19, cex = 0.3, col = "black")
map("county", "California", add = TRUE)
```


IDW Interpolation of FRP
Latitude
40
38
36
34
−124
−122
−120
−118
−116
Longitude
100
200
300

Kriging

```
# Lab 20 - kriging in R & HW9, HW7, Lab6, Lab7

# Prepare data and vgm model for kriging - use classical vario with mml fit
# coordinates(final_data_sp) <- ~ longitude + latitude
gridded(dense_grid) <- TRUE
vgm_frp <- vgm(
  model = "Sph",
  nugget = fire_fit_mml$nugget,
  psill = fire_fit_mml$cov.pars[1],
  range = fire_fit_mml$cov.pars[2]
)
plot(fire_classical, main = "FRP Semivariogram with MML Fit")
lines(fire_fit_mml, col = "red", lwd = 2)
```




```
# Ordinary Kriging
ok_frp <- krige(frp ~ 1,
                locations = final_data_sp,
                newdata = dense_grid,
                model = vgm_frp)
```

```
## [using ordinary kriging]
```

```
summary(ok_frp)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##                  min        max
## longitude -124.03550 -114.28550
## latitude    32.54793   41.89793
```

```
## Is projected: NA
## proj4string : [NA]
## Number of points: 36465
## Grid attributes:
##           cellcentre.offset cellsize cells.dim
## longitude        -124.01050     0.05       195
## latitude           32.57293     0.05       187
## Data attributes:
##    var1.pred         var1.var
##  Min.   : 34.67   Min.   :217864
##  1st Qu.: 63.47   1st Qu.:227323
##  Median : 69.08   Median :228594
##  Mean   : 69.98   Mean   :227803
##  3rd Qu.: 69.68   3rd Qu.:228810
##  Max.   :341.63   Max.   :228814
```

```
cv_ok <- krige.cv(frp ~ 1, final_data_sp, model = vgm_frp)
PRESS_ok <- sum(cv_ok$residual^2, na.rm = TRUE)

# Log-Ordinary Kriging
final_data_sp$log_frp <- log(final_data_sp$frp + 0.001) # small constant bc 0s result in -inf
vgm_log <- variogram(log_frp ~ 1, final_data_sp)
v_model_log <- fit.variogram(vgm_log, model = vgm("Sph"))
```

```
## Warning in fit.variogram(vgm_log, model = vgm("Sph")): No convergence after 200
## iterations: try different initial values?
```

```
log_ok_frp <- krige(log_frp ~ 1,
                    locations = final_data_sp,
                    newdata = dense_grid,
                    model = v_model_log)
```

```
## [using ordinary kriging]
```

```
summary(log_ok_frp)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##                  min        max
## longitude -124.03550 -114.28550
## latitude    32.54793   41.89793
## Is projected: NA
## proj4string : [NA]
## Number of points: 36465
## Grid attributes:
##           cellcentre.offset cellsize cells.dim
## longitude        -124.01050     0.05       195
## latitude           32.57293     0.05       187
## Data attributes:
##    var1.pred        var1.var
##  Min.   :2.450   Min.   :3.259
##  1st Qu.:2.771   1st Qu.:3.383
##  Median :3.161   Median :3.457
##  Mean   :3.136   Mean   :3.481
##  3rd Qu.:3.336   3rd Qu.:3.562
##  Max.   :4.752   Max.   :3.831
```

```r
cv_log_ok <- krige.cv(log_frp ~ 1, final_data_sp, model = v_model_log)
PRESS_log_ok <- sum(cv_log_ok$residual^2, na.rm = TRUE)


# Simple Kriging
mean_frp <- mean(final_data_sp$frp)
sk_frp <- krige(frp ~ 1,
                locations = final_data_sp,
                newdata = dense_grid,
                model = vgm_frp,
                beta = mean_frp)
```

```
## [using simple kriging]
```

```r
summary(sk_frp)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##                  min        max
## longitude -124.03550 -114.28550
## latitude    32.54793   41.89793
## Is projected: NA
## proj4string : [NA]
## Number of points: 36465
## Grid attributes:
##           cellcentre.offset cellsize cells.dim
## longitude        -124.01050     0.05       195
## latitude           32.57293     0.05       187
## Data attributes:
##    var1.pred         var1.var
##  Min.   : 68.55   Min.   :217862
##  1st Qu.:179.53   1st Qu.:226516
##  Median :203.31   Median :227403
##  Mean   :190.31   Mean   :226783
##  3rd Qu.:208.40   3rd Qu.:227517
##  Max.   :346.06   Max.   :227518
```

```r
cv_sk <- krige.cv(frp ~ 1, final_data_sp, model = vgm_frp, beta = mean_frp)
PRESS_sk <- sum(cv_sk$residual^2, na.rm = TRUE)

# Log-Simple Kriging
mean_log_frp <- mean(final_data_sp$log_frp)
log_sk_frp <- krige(log_frp ~ 1,
                    locations = final_data_sp,
                    newdata = dense_grid,
                    model = v_model_log,
                    beta = mean_log_frp)
```

```
## [using simple kriging]
```

```r
summary(log_sk_frp)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##                  min        max
## longitude -124.03550 -114.28550
```

```
## latitude    32.54793   41.89793
## Is projected: NA
## proj4string : [NA]
## Number of points: 36465
## Grid attributes:
##           cellcentre.offset cellsize cells.dim
## longitude        -124.01050     0.05       195
## latitude           32.57293     0.05       187
## Data attributes:
##    var1.pred        var1.var
##  Min.   :2.500   Min.   :3.259
##  1st Qu.:2.873   1st Qu.:3.382
##  Median :3.396   Median :3.453
##  Mean   :3.306   Mean   :3.466
##  3rd Qu.:3.618   3rd Qu.:3.544
##  Max.   :4.756   Max.   :3.716
```

```
cv_log_sk <- krige.cv(log_frp ~ 1, final_data_sp, model = v_model_log, beta = mean_log_frp)
PRESS_log_sk <- sum(cv_log_sk$residual^2, na.rm = TRUE)

# Universal Kriging
uk_frp <- krige(frp ~ longitude + latitude,
                locations = final_data_sp,
                newdata = dense_grid,
                model = vgm_frp)
```

```
## [using universal kriging]
```

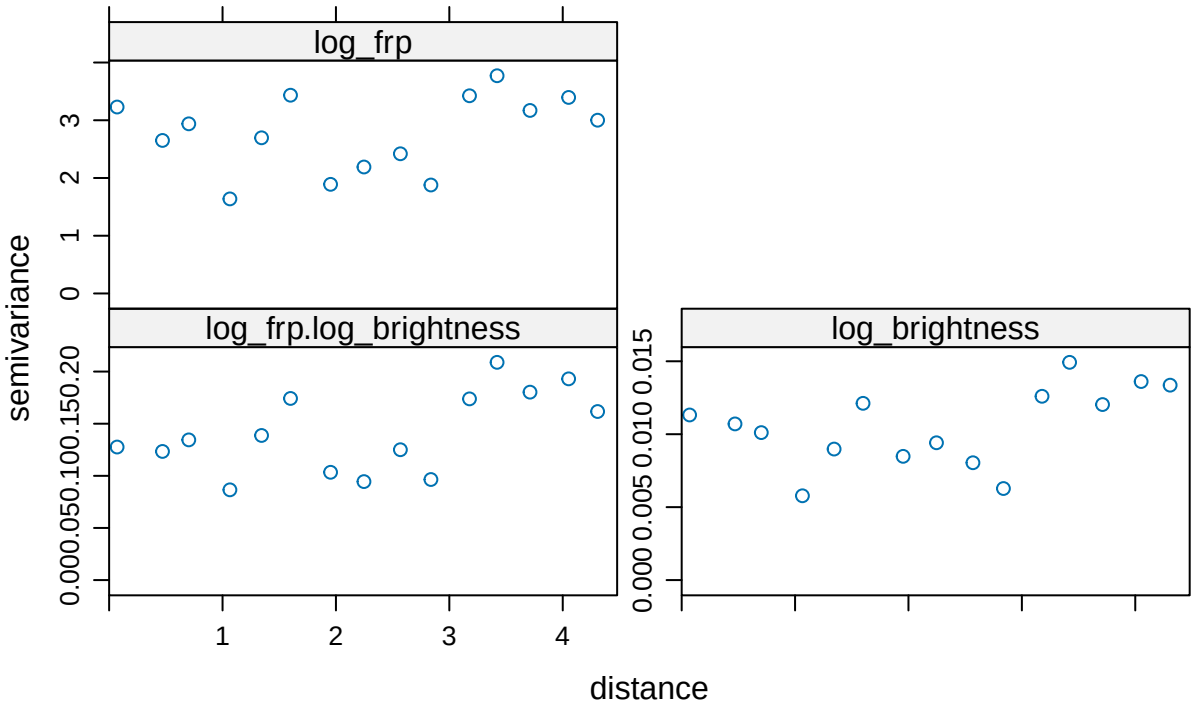
```
summary(uk_frp)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##                  min        max
## longitude -124.03550 -114.28550
## latitude    32.54793   41.89793
## Is projected: NA
## proj4string : [NA]
## Number of points: 36465
## Grid attributes:
##           cellcentre.offset cellsize cells.dim
## longitude        -124.01050     0.05       195
## latitude           32.57293     0.05       187
## Data attributes:
##    var1.pred           var1.var
##  Min.   :-98.8372   Min.   :217867
##  1st Qu.: -0.5875   1st Qu.:228600
##  Median : 48.0076   Median :232758
##  Mean   : 54.8810   Mean   :235525
##  3rd Qu.:103.2404   3rd Qu.:239936
##  Max.   :343.8748   Max.   :280216
```

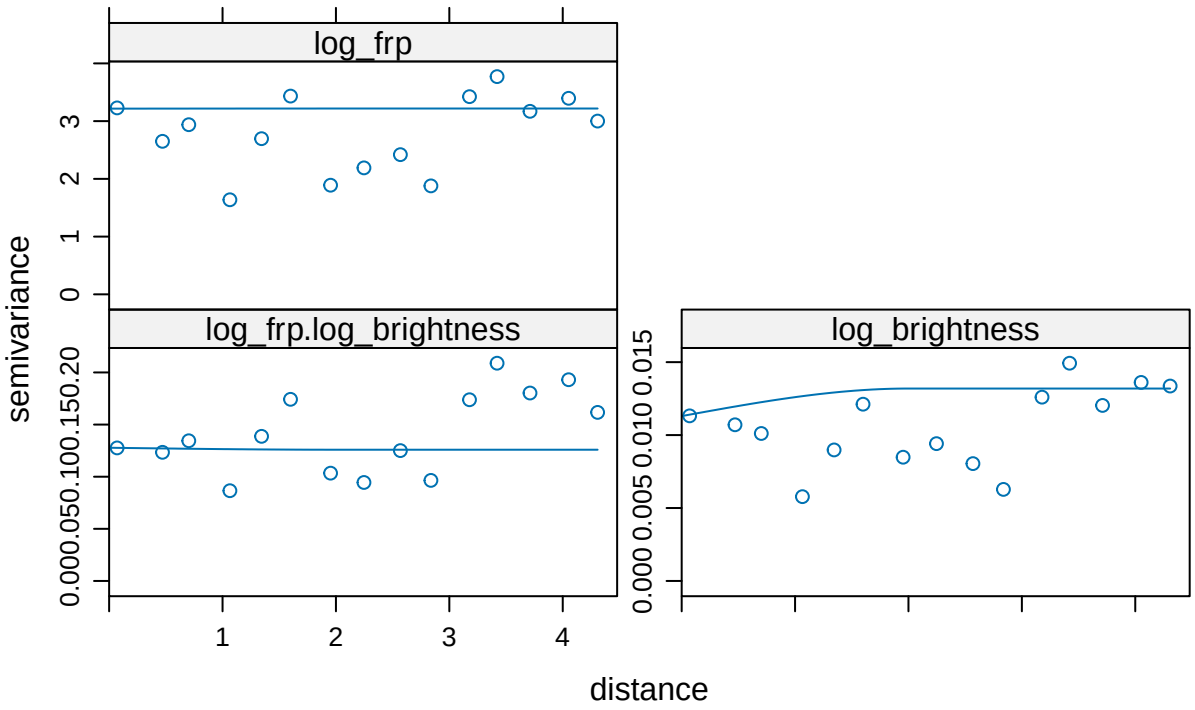
```
uk_cv <- krige.cv(frp ~ longitude + latitude, final_data_sp, model = vgm_frp)
PRESS_uk <- sum(uk_cv$residual^2, na.rm = TRUE)


# Cokriging -- HW9
```

```r
final_df <- as.data.frame(final_data)
final_df$log_frp <- log(final_df$frp + 0.001)
final_df$log_brightness <- log(final_df$brightness + 0.001)
final_df$log_bright_t31 <- log(final_df$bright_t31 + 0.001)
final_df$log_confidence <- log(final_df$confidence + 0.001)
final_df$log_scan <- log(final_df$scan + 0.001)
final_df$daynight_num <- ifelse(final_df$daynight == "D", 0, 1)
# coordinates(final_df) <- ~ longitude + latitude

# gstat objects
g <- gstat(id = "log_frp", formula = log_frp ~ 1, locations = ~ longitude + latitude, data = final_df)
g <- gstat(g, id = "log_brightness", formula = log_brightness ~ 1, locations = ~ longitude + latitude, c
# g <- gstat(g, id = "log_bright_t31", formula = log_bright_t31 ~ 1, locations = ~ longitude + latitude
# g <- gstat(g, id = "log_confidence", formula = log_confidence ~ 1, locations = ~ longitude + latitude
# g <- gstat(g, id = "log_scan", formula = log_scan ~ 1, locations = ~ longitude + latitude, data = fin
# g <- gstat(g, id = "daynight_num", formula = daynight_num ~ 1, locations = ~ longitude + latitude, da
# -- Had to remove predictors for cokriging to work. Kept frp(target) and brightness(predictor)

# variogram
vario_mult <- variogram(g)
plot(vario_mult, main = "Cokriging Sample Variograms for FRP and Predictors")
```

Cokriging Sample Variograms for FRP and Predictors




```r
# Fit model
init_model <- vgm(psill = 100, model = "Sph", range = 2, nugget = 50)
frp_fit <- fit.lmc(vario_mult, g, model = init_model)
plot(vario_mult, frp_fit, main = "Fitted LMC for FRP Cokriging")
```




```
# Predictions
ck_pred <- predict(frp_fit, newdata = dense_grid, set = list(nocheck = 1))
```

```
## Linear Model of Coregionalization found. Good.
## [using ordinary cokriging]
```

```
summary(frp_fit)
```

```
##           Length Class   Mode
## data      2      -none-  list
## model     3      -none-  list
## locations 2      formula call
## call      6      -none-  call
```

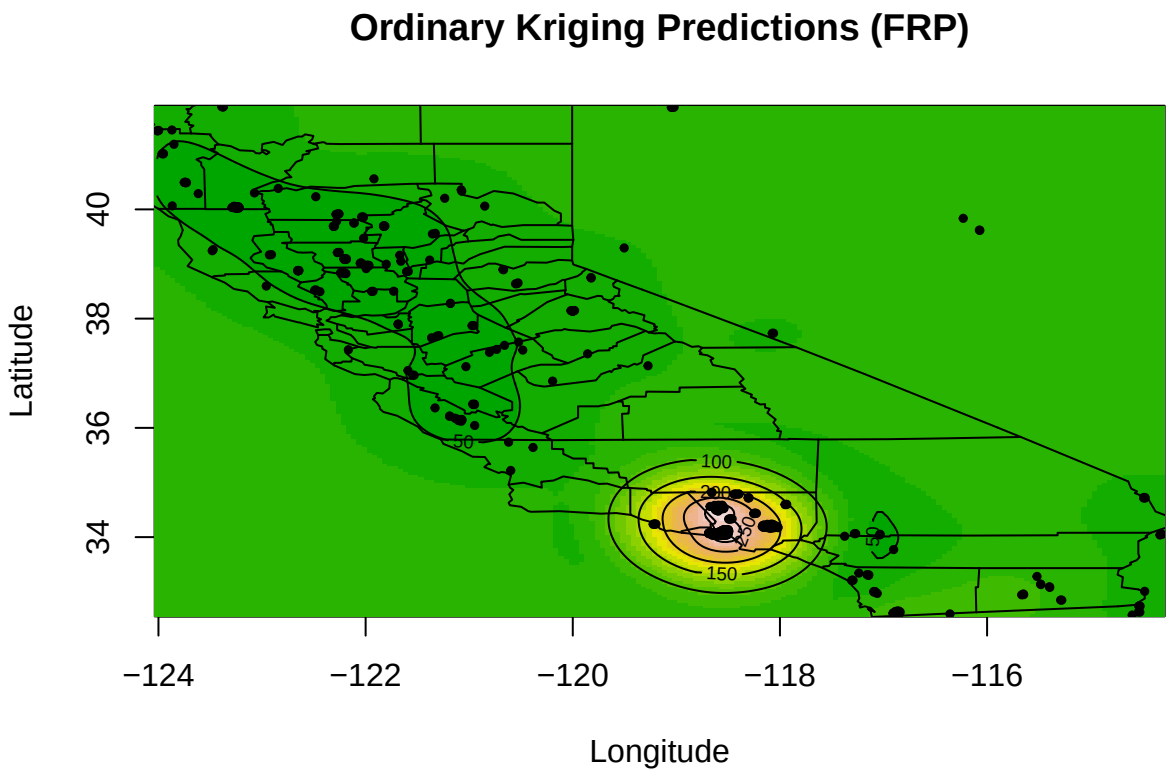
```
invisible(capture.output({cv_ck <- gstat.cv(frp_fit, verbose = FALSE)}))
PRESS_ck <- sum(cv_ck$residual^2, na.rm = TRUE)


#  PRESS Comparison Table
press_table <- data.frame(
  Method = c("OK", "Log-OK", "SK", "Log-SK", "UK", "Cokriging"),
  PRESS = c(PRESS_ok, PRESS_log_ok, PRESS_sk, PRESS_log_sk, PRESS_uk, PRESS_ck)
)
press_table
```

```
##      Method        PRESS
## 1        OK 1.637017e+08
## 2    Log-OK 1.641774e+03
## 3        SK 1.664646e+08
## 4    Log-SK 1.644158e+03
```

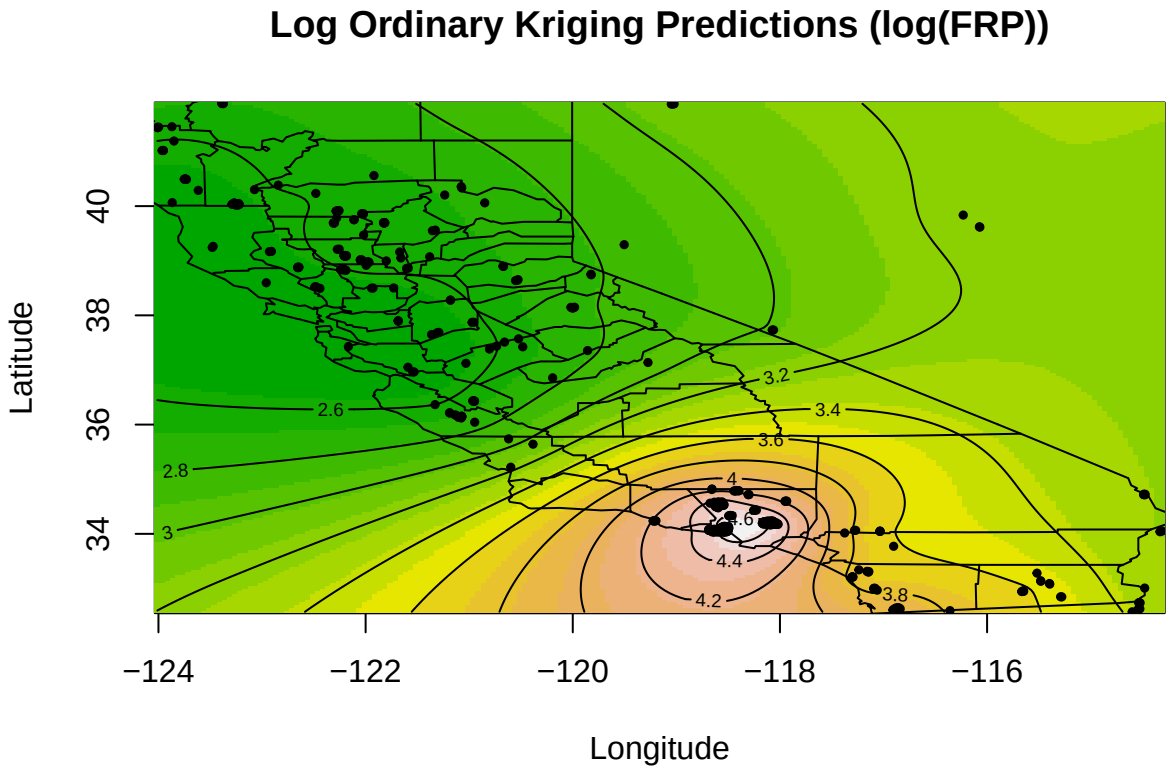
```
## 5        UK 1.638179e+08
## 6 Cokriging 1.012284e+03
```

Raster Maps

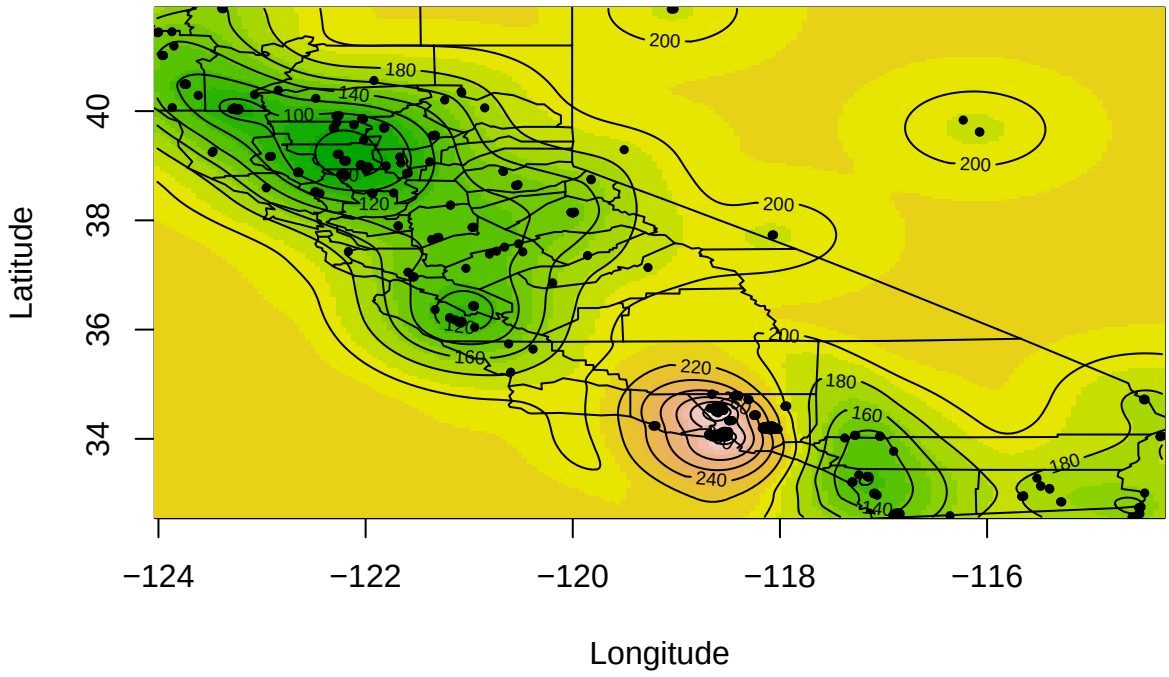
```
# Raster Map w/ contours
x_seq <- unique(dense_grid$longitude)
y_seq <- unique(dense_grid$latitude)

# Ordinary Kriging
ok_mat <- matrix(ok_frp$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x = x_seq, y = y_seq, z = ok_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "Ordinary Kriging Predictions (FRP)")
contour(x = x_seq, y = y_seq, z = ok_mat, add = TRUE, nlevels = 10)
points(final_data_sp$longitude, final_data_sp$latitude, pch = 19, cex = 0.5)
map("county", "California", add = TRUE)
```



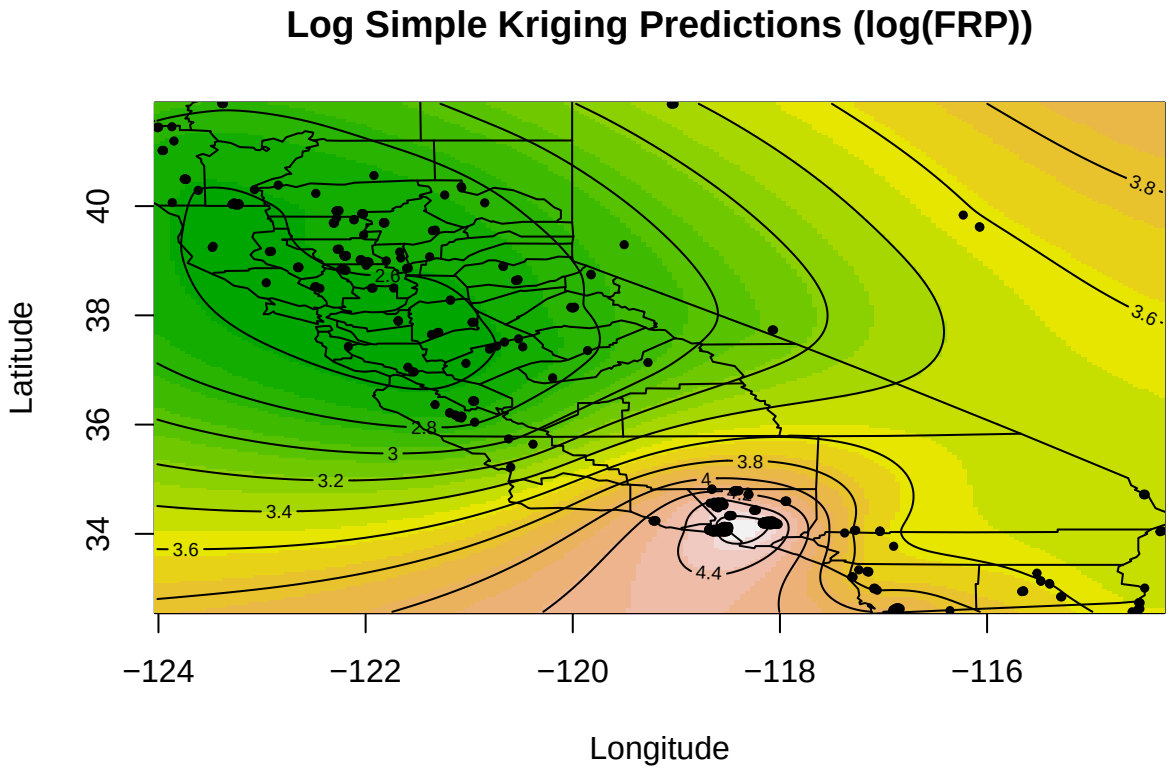

```
# Log-Ordinary Kriging
log_ok_df <- as.data.frame(log_ok_frp)
log_ok_mat <- matrix(log_ok_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x = x_seq, y = y_seq, z = log_ok_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "Log Ordinary Kriging Predictions (log(FRP))")
contour(x = x_seq, y = y_seq, z = log_ok_mat, add = TRUE, nlevels = 10)
points(final_data_sp$longitude, final_data_sp$latitude, pch = 19, cex = 0.5)
map("county", "California", add = TRUE)
```



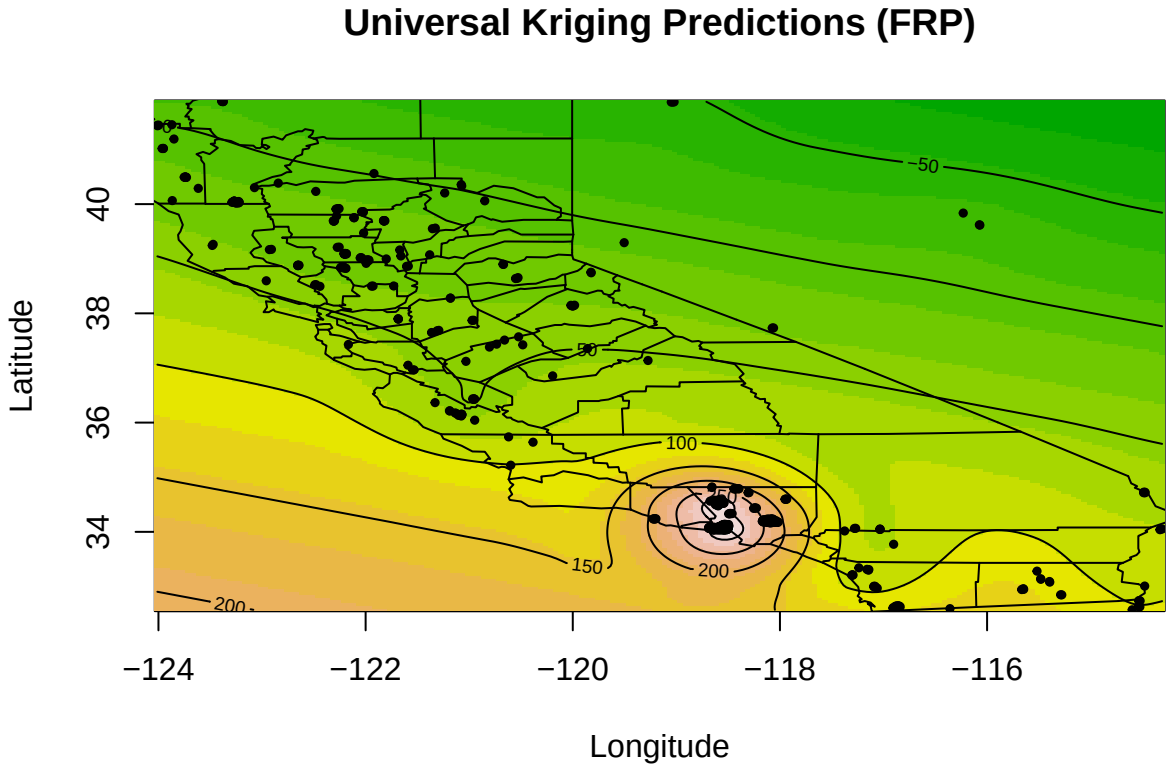

```r
# Simple Kriging
sk_df <- as.data.frame(sk_frp)
sk_mat <- matrix(sk_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x = x_seq, y = y_seq, z = sk_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "Simple Kriging Predictions (FRP)")
contour(x = x_seq, y = y_seq, z = sk_mat, add = TRUE, nlevels = 10)
points(final_data_sp$longitude, final_data_sp$latitude, pch = 19, cex = 0.5)
map("county", "California", add = TRUE)
```



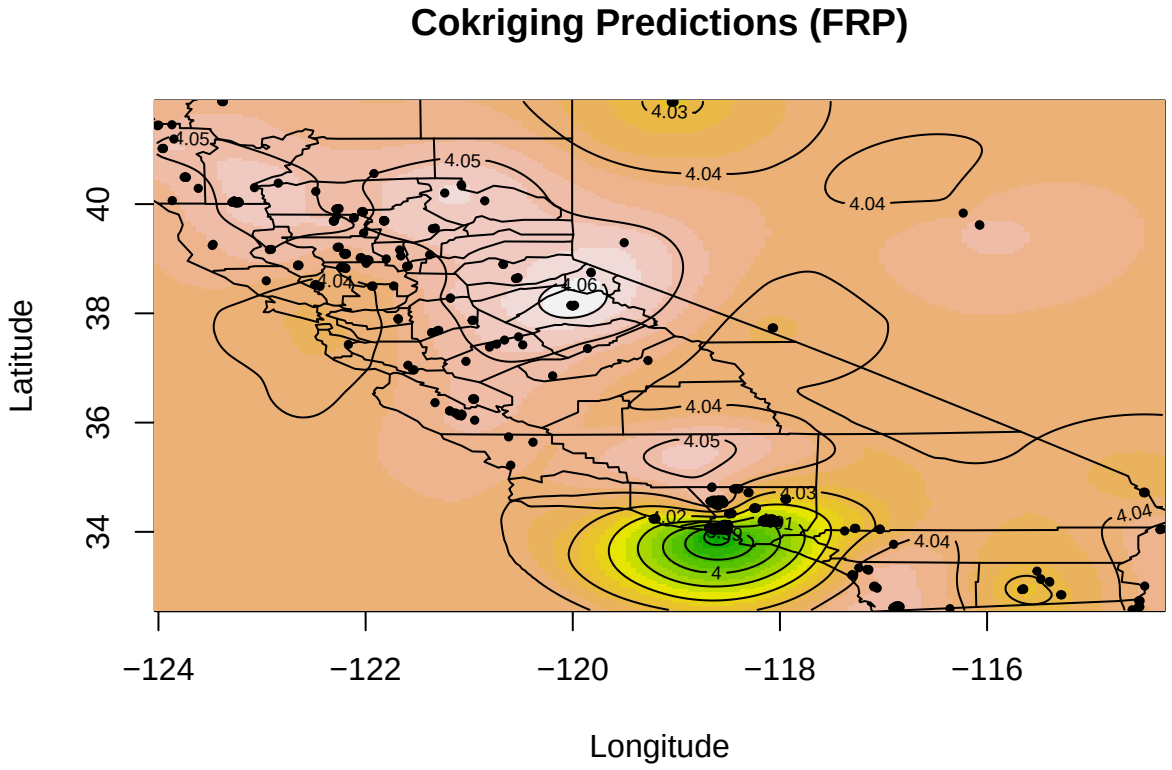

```r
# Log-Simple Kriging
log_sk_df <- as.data.frame(log_sk_frp)
log_sk_mat <- matrix(log_sk_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x = x_seq, y = y_seq, z = log_sk_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "Log Simple Kriging Predictions (log(FRP))")
contour(x = x_seq, y = y_seq, z = log_sk_mat, add = TRUE, nlevels = 10)
points(final_data_sp$longitude, final_data_sp$latitude, pch = 19, cex = 0.5)
map("county", "California", add = TRUE)
```




```
# Universal Kriging
uk_df <- as.data.frame(uk_frp)
uk_mat <- matrix(uk_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x = x_seq, y = y_seq, z = uk_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "Universal Kriging Predictions (FRP)")
contour(x = x_seq, y = y_seq, z = uk_mat, add = TRUE, nlevels = 10)
points(final_data_sp$longitude, final_data_sp$latitude, pch = 19, cex = 0.5)
map("county", "California", add = TRUE)
```




```
# Cokriging
ck_df <- as.data.frame(ck_pred)
ck_mat <- matrix(ck_df$log_frp.pred,
                 nrow = length(x_seq),
                 ncol = length(y_seq))
image(x = x_seq, y = y_seq, z = ck_mat,
      col = terrain.colors(20),
      xlab = "Longitude", ylab = "Latitude",
      main = "Cokriging Predictions (FRP)")
contour(x = x_seq, y = y_seq, z = ck_mat, add = TRUE, nlevels = 10)
points(final_data_sp$longitude, final_data_sp$latitude, pch = 19, cex = 0.5)
map("county", "California", add = TRUE)
```


Cokriging Predictions (FRP)
Latitude
Longitude
40
38
36
34
−124
−122
−120
−118
−116
4.05
4.03
4.04
4.06
4.02
4.01
4