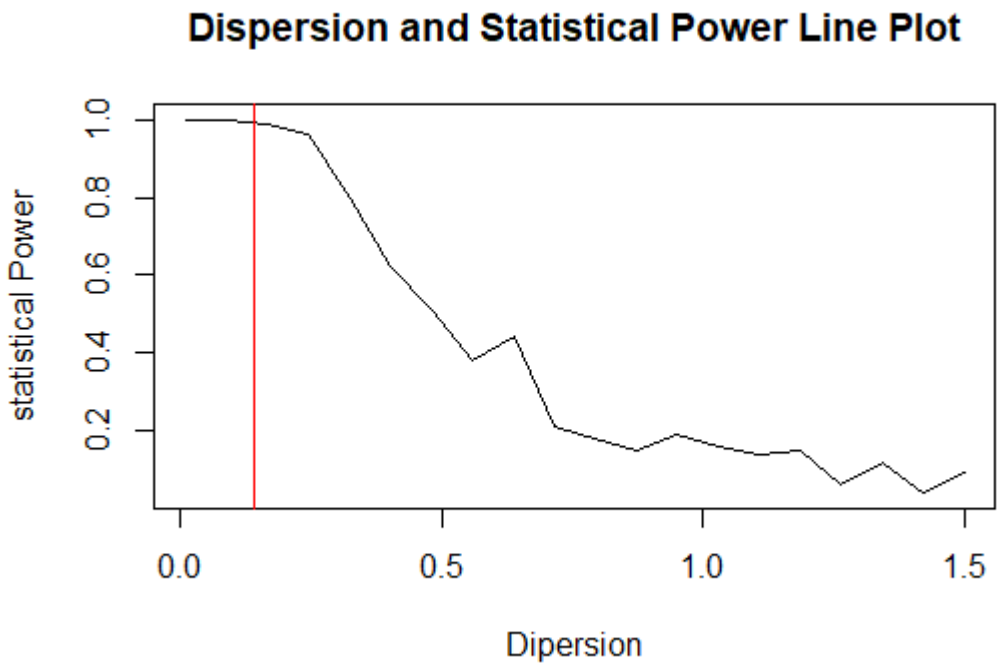
Keegan Moynahan

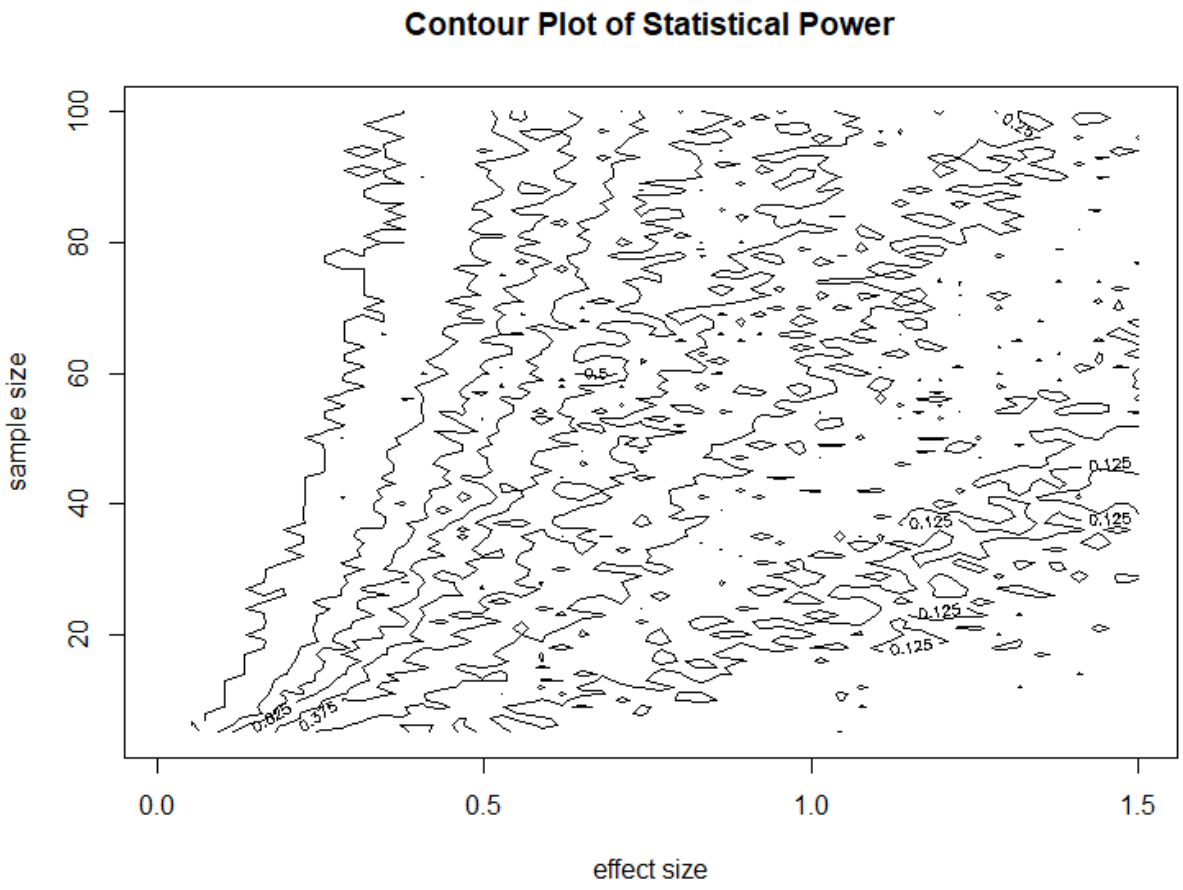
Lab 11

1)



2) Statistical power decreases as population dispersion increases because the model becomes less confident with more noise. The easiest way to look at it in my opinion is to view dispersion as the sd (which it is). When the sd increases, the statistical power is going to become less confident because of the noise growing.

3)



4) This contour plot shows that with smaller sample and smaller effect sizes the statistical power is low (beginning of plot). As both the effect size and sample size become larger you can see that the statistical power becomes higher (closer contours). In this contour plot there seems to be a "sweet spot" where the effect size and sample size create the highest power and then towards the 1.5 of effect size the power seems to decrease. I would say for this contour plot the areas that have the highest statistical power are roughly 0.2-1 (effect size). I would also say that a smaller population dispersion seems to create a high power. The area of sample size (start-40) has the closest lines. This shows that a lower population dispersion makes for a higher power value.
5) file:///E:/eco602_eco634/docs/sim_3_dat_plot.html will submit separately to Moodle.
6) I could use the information in my plot to design an experiment by viewing that the power is high when the beta value is low. As the beta value increases the power decreases. I could use this as a hypothesis and use different beta values to test if this is true for my plot. I could set it up to accept the hypothesis or reject it.

Population dispersion analysis & Line plot:

```
#lab question 1----

alpha = 0.05
n_sims = 100
p_vals = numeric(n_sims)

# What was the observed standard deviation?
sd_obs

# specify the number of different standard deviation values to simulate:
n_sds = 20
pop_sds = seq(from = 0.01, to = 1.5, length.out = n_sds)

pop_sd_powers = numeric(n_sds)

for(j in 1:length(pop_sds))
{
 pop_sd_j = pop_sds[j]
 for(i in 1:n_sims)
 {
  fit_sim = linear_sim_fit(
    x = birdhab$ls,
    y_int = int_obs,
    slope = slope_obs,
    st_dev = pop_sd_j
    )
  p_vals[i] =  summary(fit_sim)$coefficients[2, 'Pr(>|t|)']
 }
```

```
  pop_sd_powers[j] = sum(p_vals < alpha) / n_sims
}

sim_output_dispersion = data.frame(
  sd = pop_sds,
  power = pop_sd_powers)

# You should save your simulation results so you don't have to run it every time.
save(
  sim_output_dispersion,
  file = here::here("data", "lab_ll_dat_dispersion_sim.RData"))

# Line plot of standard deviation (x-axis) and statistical power (y-axis)
plot(pop_sds, pop_sd_powers, xlab = "Dipersion", ylab = "statistical Power", type = "line", main =
"Dispersion and Statistical Power Line Plot")

# Add a dotted vertical red line at the observed population standard deviation value.
abline(v = sd_obs, col = "red")
```

Population Dispersion and Sample Analysis & Contour plot & interactive plot:

```
# lab question 2 ----


alpha = 0.05

# Start with a small number
n_sims = 100
p_vals = numeric(n_sims)

# What was the observed standard deviation?
```

```
sd_obs

# specify the number of different standard deviation values to simulate:
# Start with a small number
n_sds = 50
pop_sds = seq(from = 0.01, to = 1.5, length.out = n_sds)

# The maximum x value in the simulation.
# Use the maximum observed x-value in the data
max_x = max(birdhab$ls)



pop_sd_powers = numeric(n_sds)

sample_sizes = seq(5, 100)

sim_output_3 = matrix(nrow = length(pop_sds), ncol = length(sample_sizes))

for(k in 1:length(pop_sds))
{
 pop_sd_k = pop_sds[k]

 for(j in 1:length(sample_sizes))
 {
  x_vals = seq(0, max_x, length.out = sample_sizes[j])

  for(i in 1:n_sims)
  {
   fit_sim = linear_sim_fit(
```

```
      x = x_vals,
      y_int = int_obs,
      slope = slope_obs,
      st_dev = pop_sd_k
    )
    summary(fit_sim)
    p_vals[i] = summary(fit_sim)$coefficients[2, 'Pr(>|t|)']
  }

  sim_output_3[k, j] = sum(p_vals < alpha) / n_sims
 }
 print(paste0("Testing standard deviation ", k, " of ", n_sds))
}

image(sim_output_3)

sim_3_dat =
 list(
  power       = sim_output_3,
  sample_size = sample_sizes,
  pop_sd      = pop_sds
  )


# You should save your simulation results so you don't have to run it every time.
save(
 sim_3_dat,
 file = here::here("data", "lab_ll_sim_output_dispersion_n_1000.RData"))
```

```
#contour
contour(
  x = sim_3_dat$pop_sd,
  y = sim_3_dat$sample_size,
  z = sim_3_dat$power,
  xlab = "effect size",
  ylab = "sample size",
  main = "Contour Plot of Statistical Power",
  levels = seq(0, 1, length.out = 9),
  drawlabels = TRUE,
  # method = "simple")
  method = "edge")

# Persp3d
library("rgl")
persp3d(
  x = sim_3_dat$pop_sd,
  y = sim_3_dat$sample_size,
  z = sim_3_dat$power,
  xlab = "beta", ylab = "n", zlab = "power",
  col = 'red',
  theta = 30, phi = 30, expand = .75,
  ticktype = 'detailed')
require(htmlwidgets)
saveWidget(
  rglwidget(),
  file = here(
    "docs",
    "sim_3_dat_plot.html"),
```

```
  selfcontained = TRUE
)
```