Flume: Full Analysis

(adapted from Jarad Niemi - Soilpad Analysis)

Jessica Nelson

2022-03-24

Contents

```
knitr::opts_chunk$set(echo = TRUE,
                      cache = TRUE,
                      fig.width = 12,
                      fig.height = 12)

library("lme4")
```

```
## Loading required package: Matrix
```

```
library("lmerTest")
```

```
##
## Attaching package: 'lmerTest'

## The following object is masked from 'package:lme4':
##
##     lmer

## The following object is masked from 'package:stats':
##
##     step
```

```
library("tidyverse"); theme_set(theme_bw())
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
```

```
library("emmeans")
library("ggResidpanel")
library("data.table")
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library("stringr")


options(width = 120)

dir.create("fig", showWarnings = FALSE)
```

```
sessionInfo()
```

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252    LC_MONETARY=Englis
## [4] LC_NUMERIC=C                           LC_TIME=English_United States.1252
##
```

```
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] data.table_1.14.2  ggResidpanel_0.3.0 emmeans_1.7.2      forcats_0.5.1      stringr_1.4.0
##  [7] purrr_0.3.4        readr_2.1.2        tidyr_1.2.0        tibble_3.1.6       ggplot2_3.3.5
## [13] lmerTest_3.1-3     lme4_1.1-28        Matrix_1.4-1
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2          viridisLite_0.4.0   jsonlite_1.8.0      splines_4.1.3       modelr_0.1.8
##  [6] assertthat_0.2.1    cellranger_1.1.0    robustbase_0.93-9   yaml_2.3.5          numDeriv_2016.8
## [11] pillar_1.7.0        backports_1.4.1     lattice_0.20-45     glue_1.6.2          digest_0.6.29
## [16] rvest_1.0.2         minqa_1.2.4         colorspace_2.0-3    cowplot_1.1.1       htmltools_0.5.2
## [21] pkgconfig_2.0.3     broom_0.7.12        haven_2.4.3         xtable_1.8-4        mvtnorm_1.1-3
## [26] scales_1.1.1        tzdb_0.2.0          generics_0.1.2      ellipsis_0.3.2      withr_2.5.0
## [31] lazyeval_0.2.2      cli_3.2.0           magrittr_2.0.1      crayon_1.5.0        readxl_1.3.1
## [36] estimability_1.3    evaluate_0.15       fs_1.5.2            fansi_1.0.2         nlme_3.1-155
## [41] MASS_7.3-55         xml2_1.3.3          tools_4.1.3         hms_1.1.1           lifecycle_1.0.1
## [46] plotly_4.10.0       munsell_0.5.0       reprex_2.0.1        qqplotr_0.0.5       compiler_4.1.3
## [51] rlang_1.0.2         grid_4.1.3          nloptr_2.0.0        rstudioapi_0.13     htmlwidgets_1.5
## [56] rmarkdown_2.13      boot_1.3-28         gtable_0.3.0        DBI_1.1.2           R6_2.5.1
## [61] lubridate_1.8.0     knitr_1.37          fastmap_1.1.0       utf8_1.2.2          stringi_1.7.6
## [66] Rcpp_1.0.8.3        vctrs_0.3.8         DEoptimR_1.0-10     dbplyr_2.1.1        tidyselect_1.1.
## [71] xfun_0.30
```

Read in data

```
library("tidyverse")

flume <- read_csv("../data/tidy/flume_event_data612_UPDATE.csv") %>%
  mutate(Year = factor(Year)) %>%
  subset(subtreatment != 'grass strip') %>%
  subset(SiteID != 'MCN') %>%
  subset(subset=!(SiteID=="RHO" & Year == 2016)) %>%
  subset(subset=!(SiteID=="RHO" & Year == 2017))
```

```
## Rows: 432 Columns: 19
## -- Column specification ------------------------------------------------------------------------
## Delimiter: ","
## chr  (7): SiteID, subtreatment, Treatment, sampleID, random, crop, f_loc
## dbl (12): precipitation, rain_time, rf_event, sample_event, ro_event, Year, flow_time, flow, tss_sum
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
flume_sum <- flume %>%
  group_by(Treatment, Year, SiteID, sample_event, tss_sum, crop) %>%
  summarize(tss_load = tss_sum,
            ln_tss_load = log(tss_load+0.000198)) %>%
  distinct()
```

```
## `summarise()` has grouped output by 'Treatment', 'Year', 'SiteID', 'sample_event', 'tss_sum', 'crop'
## using the `.groups` argument.
```

```
ppt_sum <- flume %>%
  group_by(Treatment, Year, SiteID, sample_event, crop) %>%
  summarize(ppt_sum = sum(precipitation)) %>%
  ungroup() %>%
  filter(!duplicated(cbind(Year, SiteID, sample_event)))
```

```
## `summarise()` has grouped output by 'Treatment', 'Year', 'SiteID', 'sample_event'. You can override u
## argument.
```

```
sample_anova <- flume_sum %>%
  filter(!is.na(tss_sum)) %>%
  select(Year, SiteID, Treatment, sample_event, tss_sum, crop) %>%
  group_by(SiteID, Year, Treatment, sample_event, crop) %>%
    summarize(tss_load = sum(tss_sum)) %>%
  ungroup() %>%
  select(Year, SiteID, Treatment, sample_event, tss_load, crop) %>%
  pivot_wider(names_from = Treatment, values_from = tss_load)
```

```
## `summarise()` has grouped output by 'SiteID', 'Year', 'Treatment', 'sample_event'. You can override u
## argument.
```

```
pivot_sample <- sample_anova %>%
  inner_join(ppt_sum,by=c("SiteID", "Year", "sample_event", "crop")) %>%
  filter(!is.na(strips)) %>%
  mutate(ln_ppt = log(ppt_sum),
         diff = strips-control,
         ln_diff = log(abs(diff)+0.00165),
         ln_ctl = log(control+0.0053),
         ln_trt = log(strips+0.0104)) %>%
  subset(select = -c(Treatment))

long_load <- pivot_sample %>%
  gather(Treatment, tss_load, control:strips) %>%
  arrange(Treatment, tss_load) %>%
  filter(!is.na(diff)) %>%
  select(SiteID, Treatment, Year, sample_event, tss_load, diff, ppt_sum, crop)
```

```
rf_ro_pivot <- long_load %>%
  mutate(random = (ifelse(SiteID == 'ARM', 'NR',
  ifelse(SiteID == 'EIA', 'R',
  ifelse(SiteID == 'MCN', 'R',
  ifelse(SiteID == 'HOE', 'NR',
  ifelse(SiteID == 'MAR', 'NR',
  ifelse(SiteID == 'RHO', 'R',
  ifelse(SiteID == 'WHI', 'NR',
  ifelse(SiteID == 'WOR', 'R', 0)))))))))

long_load <- long_load %>%
  mutate(random = (ifelse(SiteID == 'ARM', 'NR',
  ifelse(SiteID == 'EIA', 'R',
  ifelse(SiteID == 'MCN', 'R',
  ifelse(SiteID == 'HOE', 'NR',
  ifelse(SiteID == 'MAR', 'NR',
  ifelse(SiteID == 'RHO', 'R',
  ifelse(SiteID == 'WHI', 'NR',
  ifelse(SiteID == 'WOR', 'R', 0))))))))))
```

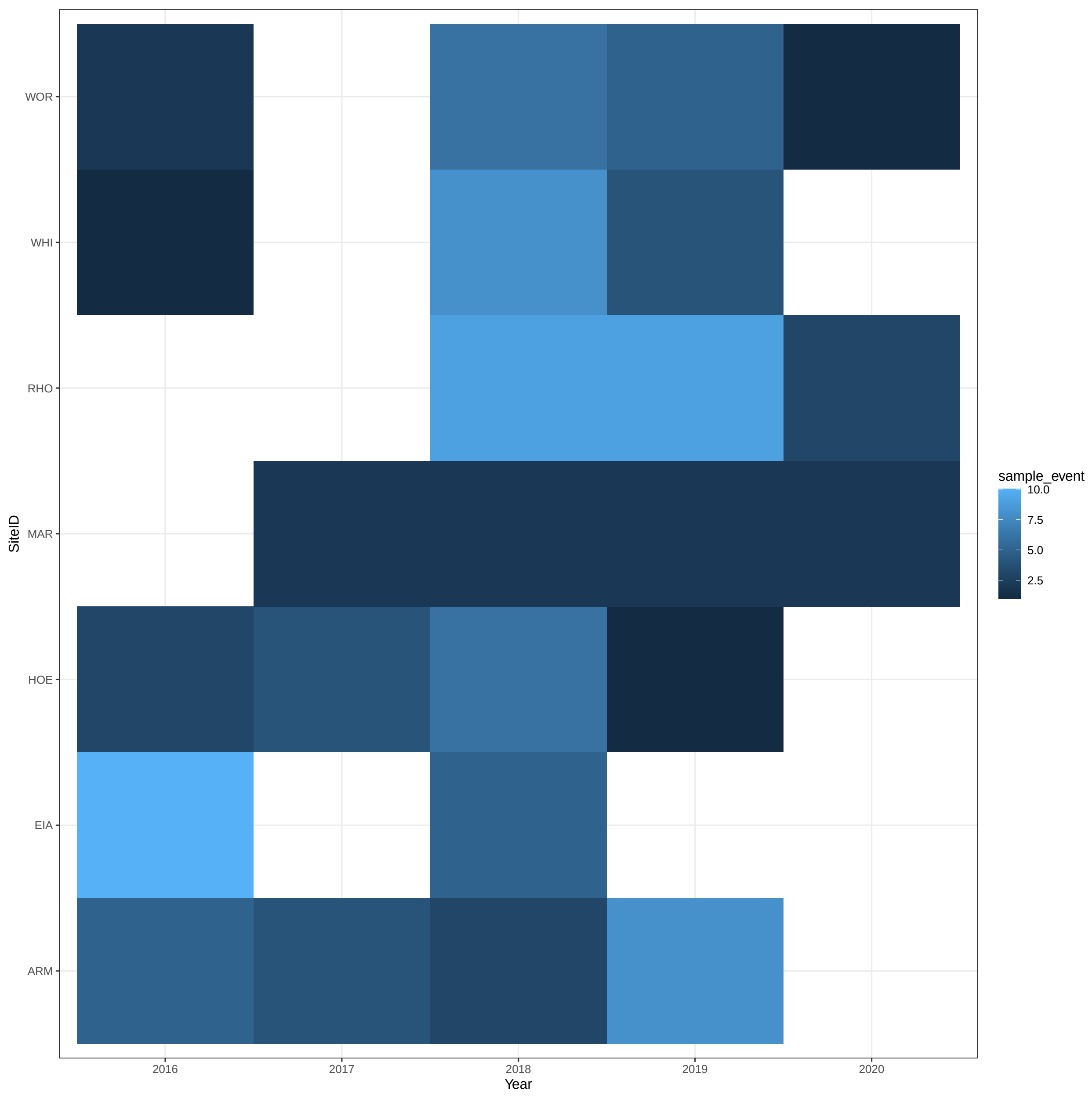
```
full_df <- rf_ro_pivot %>%
  inner_join(ppt_sum,by=c("SiteID", "Year", "sample_event","crop")) %>%
  drop_na(tss_load) %>%
  mutate(ppt_sum = ppt_sum.x,
         ln_ppt = log(ppt_sum),
         ln_tss_load = log(tss_load+0.0053),
         Treatment = Treatment.x) %>%
  subset(select = -c(Treatment.y, Treatment.x, ppt_sum.x, ppt_sum.y)) %>%
  arrange(Year, SiteID, Treatment, sample_event)

save(full_df, file = "full_df.RData")
#write.csv(full_df,"D:/ISU/ResearchProject/flume_analysis/data/tidy/full_df.csv", row.names = FALSE)
```

```
load("full_df.RData")
```

Exploratory analysis

Site-year with sample event

```
site_year_rfevent <- full_df %>%
  select(SiteID, Year, sample_event) %>%
  unique()

ggplot(site_year_rfevent, aes(Year, SiteID, fill=sample_event)) +
  geom_tile()
```



Data visualization

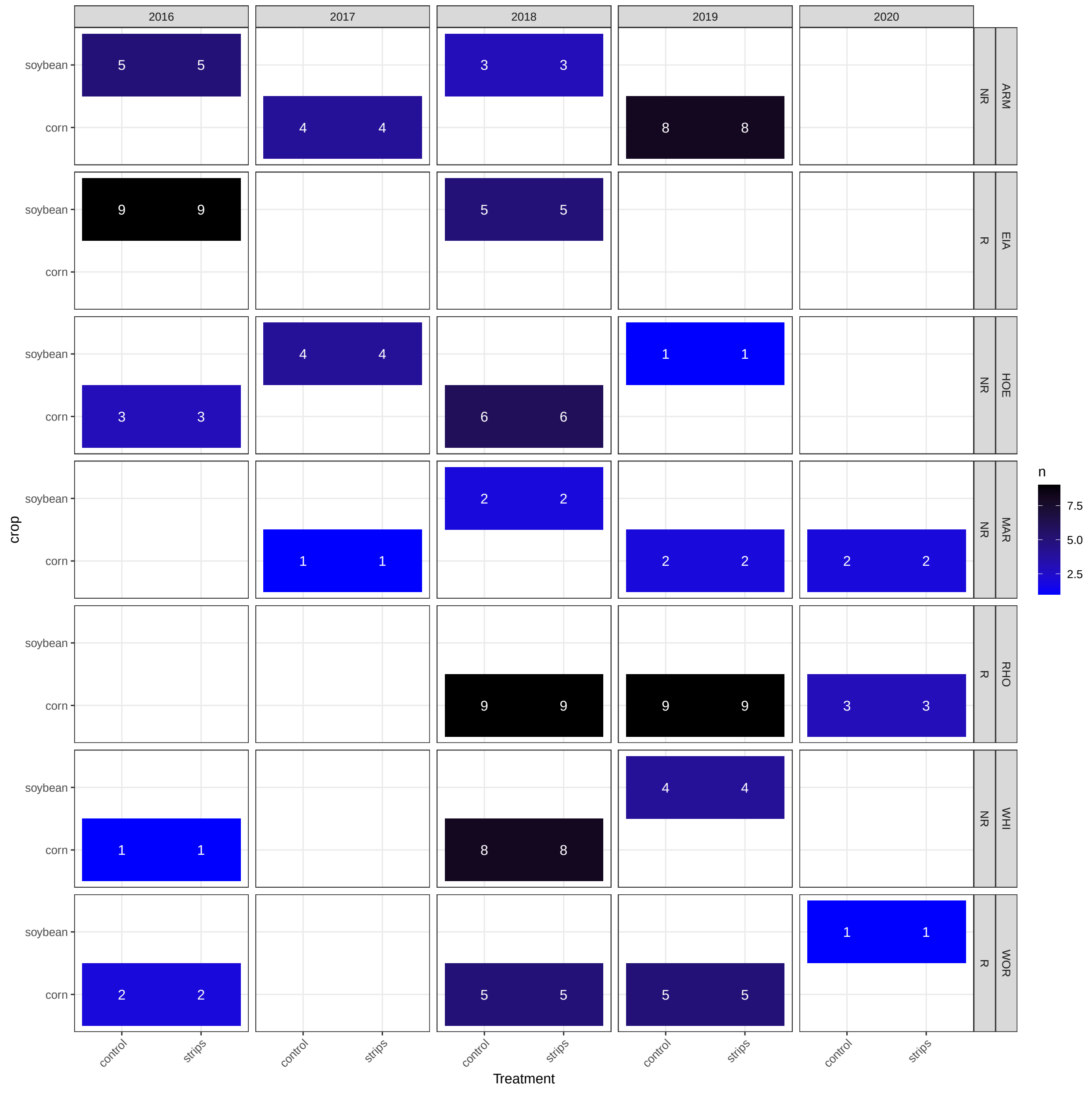
Number of samples

Calculate the number of observations for each treatment-position-year-site-time combination.

```
TSS_counts <- full_df %>%
  group_by(Year, SiteID, Treatment, crop, random) %>%
  distinct() %>%
  summarize(n = n(), .groups = "drop")
```

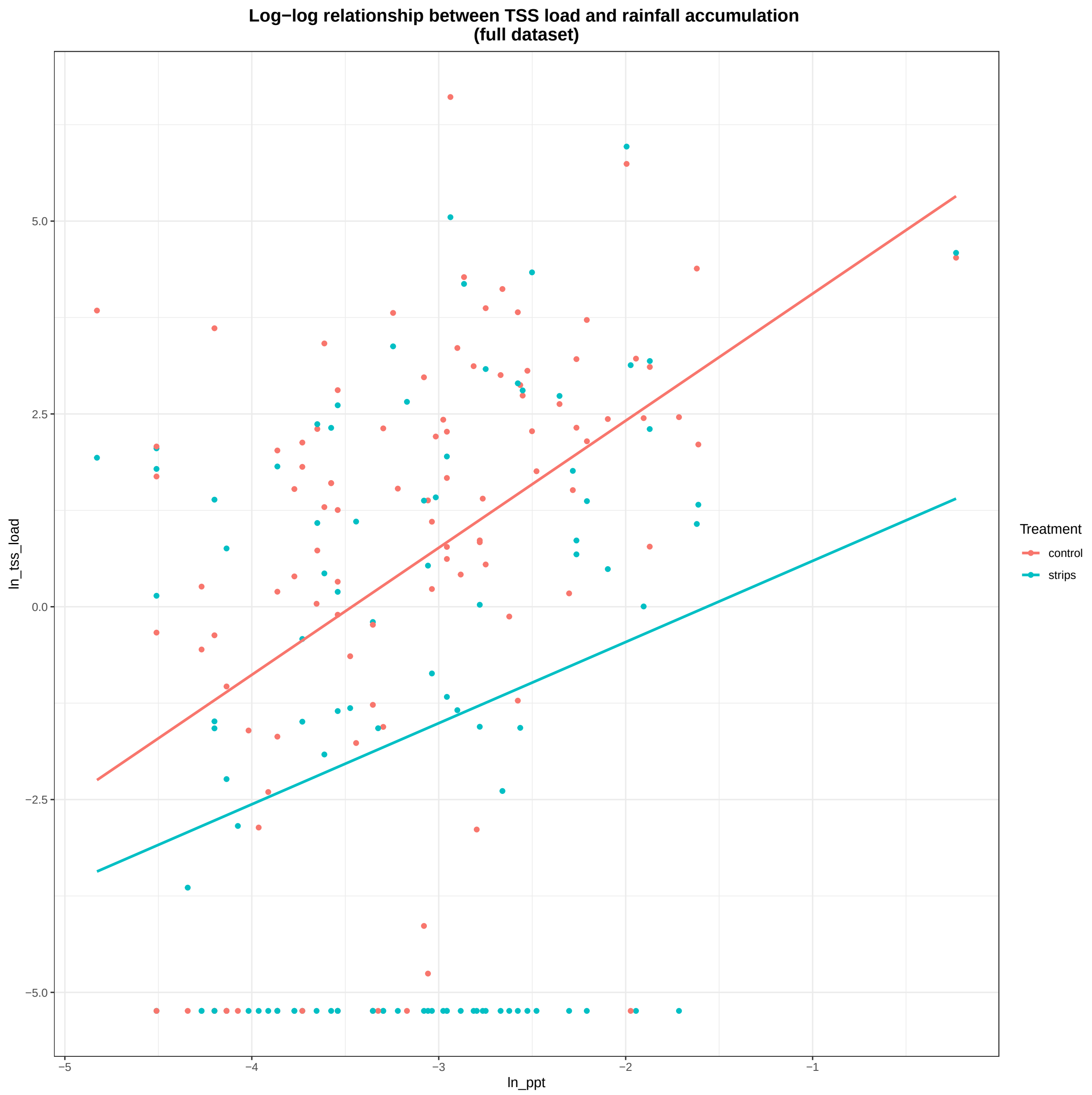
Plot the number of observations for each combination.

```r
g = ggplot(TSS_counts, aes(x = Treatment, y = crop, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n), color = "white") +
  facet_grid(SiteID + random ~ Year) +
  scale_fill_gradient(low = "blue", high = "black") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

g
```



```r
h <- ggplot(full_df, aes(x=ln_ppt, y=ln_tss_load, color=Treatment), inherit.aes = FALSE) +
  geom_point() +
  geom_smooth(method=lm, se=FALSE, fullrange=TRUE) +
  ggtitle("Log-log relationship between TSS load and rainfall accumulation \n(full dataset)") +
  theme(plot.title = element_text(size=14, face="bold",hjust = 0.5))

h
```

```
## 'geom_smooth()' using formula 'y ~ x'
```



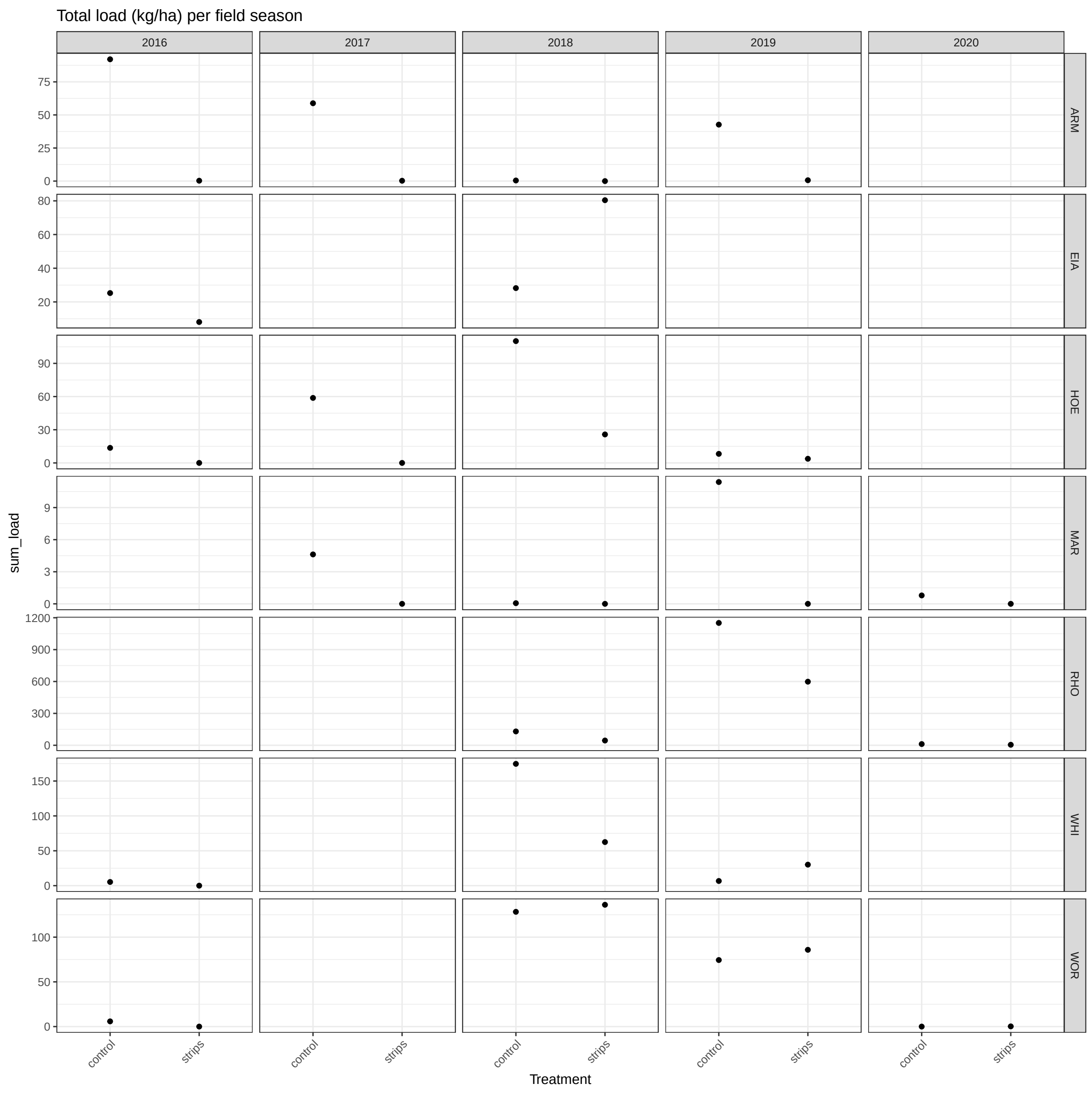
```
#ggsave("fig/randReg_ppt_load.png", h, width = 12, height = 12)
```

```
load_sum <- full_df %>%
  group_by(Year, SiteID, Treatment, crop) %>%
  summarize(sum_load              = sum(tss_load, na.rm = TRUE),
            log_load = mean(log(sum_load), na.rm = TRUE),
```

```r
            n                    = n(),
            .groups = "drop")


ppt_sum <- flume %>%
  group_by(Year, SiteID) %>%
  summarize(sum_ppt              = sum(precipitation, na.rm = TRUE),
            #log_load = mean(log(sum_load), na.rm = TRUE),
            n                    = n(),
            .groups = "drop")
```

```r
g <- ggplot(load_sum,
           aes(x = Treatment,
               y = sum_load)) +
  geom_point() +
  geom_line() +
  facet_grid(SiteID ~ Year, scales = "free_y") +
  #scale_y_log10() +
  labs(title ="Total load (kg/ha) per field season") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
g
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?
```



```
#ggsave("fig/wp_per_day_plot.png", g)
```

Average isn't realistic

```
#{r, dependson="create_sediment"} #pivot_sample %>%  #  anova_test(ln_trt ~ ln_ppt*ln_ctl)
## purr https://stackoverflow.com/questions/50702152/compare-models-via-anova-with-purrr-or-dplyr
## anova() and may need an linear model built up. #
```

Main Analyses

There are three main analyses of interest:

- confirmatory, design-based analysis

- exploratory, covariate analysis
- relationship of sediment flow to sediment loss

Confirmatory, design-based analysis

Treatment effect

```r
m_flume <- lmerTest::lmer(log(tss_load+0.0053) ~
                             (1 | SiteID) +
                             (1 | SiteID:Treatment) +
                            #(1|SiteID:Treatment:sample_event) + #removed due to singular fit

                             Treatment*ln_ppt +
                            Treatment*crop +

                            Year*Treatment,
                          data = full_df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
summary(m_flume)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']
## Formula: log(tss_load + 0.0053) ~ (1 | SiteID) + (1 | SiteID:Treatment) +
##     Treatment * ln_ppt + Treatment * crop + Year * Treatment
##    Data: full_df
##
## REML criterion at convergence: 948.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.4855 -0.5518  0.0532  0.6458  2.3193
##
## Random effects:
##  Groups           Name        Variance Std.Dev.
##  SiteID:Treatment (Intercept) 2.418    1.555
##  SiteID           (Intercept) 0.000    0.000
##  Residual                     6.214    2.493
## Number of obs: 204, groups:  SiteID:Treatment, 14; SiteID, 7
##
## Fixed effects:
##                          Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)                5.5820     1.3304 116.1012   4.196 5.35e-05 ***
## Treatmentstrips           -5.9069     1.8815 116.1012  -3.139  0.00215 **
## ln_ppt                     1.6556     0.3131 180.5163   5.288 3.55e-07 ***
## cropsoybean               -1.5253     0.6685 189.0646  -2.282  0.02362 *
## Year2017                   2.1011     1.0783 186.3242   1.948  0.05286 .
## Year2018                   0.5018     0.7683 185.2658   0.653  0.51447
## Year2019                   1.0320     0.8409 187.6670   1.227  0.22126
## Year2020                  -1.7943     1.2972 189.0783  -1.383  0.16822
## Treatmentstrips:ln_ppt    -0.6283     0.4428 180.5163  -1.419  0.15765
## Treatmentstrips:cropsoybean   1.9225     0.9454 189.0646   2.034  0.04339 *
## Treatmentstrips:Year2017      -1.5132     1.5250 186.3242  -0.992  0.32233
## Treatmentstrips:Year2018       1.3231     1.0866 185.2658   1.218  0.22492
## Treatmentstrips:Year2019       2.1215     1.1892 187.6670   1.784  0.07603 .
## Treatmentstrips:Year2020       2.6000     1.8345 189.0783   1.417  0.15804
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation matrix not shown by default, as p = 14 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
m_flume_step <- step(m_flume, reduce.random = FALSE, alpha.fixed = 0.1)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
m_flume_model <- get_model(m_flume_step)
summary(m_flume_model)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']
## Formula: log(tss_load + 0.0053) ~ (1 | SiteID) + (1 | SiteID:Treatment) +
##     Treatment + ln_ppt + crop + Year + Treatment:crop + Treatment:Year
##    Data: full_df
##
## REML criterion at convergence: 951.2
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.3490 -0.5015  0.0527  0.6269  2.2820
##
## Random effects:
##  Groups           Name        Variance Std.Dev.
##  SiteID:Treatment (Intercept) 2.397    1.548
##  SiteID           (Intercept) 0.000    0.000
##  Residual                     6.252    2.500
## Number of obs: 204, groups:  SiteID:Treatment, 14; SiteID, 7
##
## Fixed effects:
##                              Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)                    4.6199     1.1453  83.6018   4.034 0.000121 ***
## Treatmentstrips               -3.9795     1.3022  43.2396  -3.056 0.003836 **
## ln_ppt                         1.3417     0.2221 181.5700   6.042 8.42e-09 ***
## cropsoybean                   -1.5202     0.6702 189.9303  -2.268 0.024437 *
## Year2017                       2.0290     1.0802 187.4554   1.878 0.061875 .
## Year2018                       0.4806     0.7704 186.3833   0.624 0.533539
## Year2019                       0.9494     0.8412 188.8405   1.129 0.260492
## Year2020                      -1.8283     1.3005 190.1693  -1.406 0.161417
## Treatmentstrips:cropsoybean   1.9134     0.9478 189.9356   2.019 0.044907 *
## Treatmentstrips:Year2017     -1.3745     1.5259 187.4817  -0.901 0.368854
## Treatmentstrips:Year2018      1.3655     1.0893 186.4036   1.253 0.211594
## Treatmentstrips:Year2019      2.2852     1.1868 188.9185   1.925 0.055680 .
## Treatmentstrips:Year2020      2.6645     1.8389 190.1869   1.449 0.148996
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation matrix not shown by default, as p = 13 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
##https://campus.datacamp.com/courses/hierarchical-and-mixed-effects-models-in-r/linear-mixed-effect-mo
```

#"'{r design_step_model, dependson = "design_model"} #emmip(m_flume, ln_ppt ~ Treatment | Year)

##https://campus.datacamp.com/courses/hierarchical-and-mixed-effects-models-in-r/linear-mixed-effect-models?ex=7 #"'

```
trt_yr = emmeans(m_flume, pairwise ~ Treatment|Year,
                    type = "response",
                    lmer.df = "asymptotic")
confint(trt_yr)$contrasts
```

```
## Year = 2016:
##  contrast         ratio     SE  df asymp.LCL asymp.UCL
##  control / strips 19.31  23.46 Inf     1.784     209.0
##
## Year = 2017:
##  contrast         ratio     SE  df asymp.LCL asymp.UCL
##  control / strips 87.68 132.59 Inf     4.526    1698.5
##
## Year = 2018:
##  contrast         ratio     SE  df asymp.LCL asymp.UCL
##  control / strips  5.14   5.34 Inf     0.670      39.4
##
## Year = 2019:
##  contrast         ratio     SE  df asymp.LCL asymp.UCL
##  control / strips  2.31   2.57 Inf     0.263      20.4
##
## Year = 2020:
##  contrast         ratio     SE  df asymp.LCL asymp.UCL
##  control / strips  1.43   2.48 Inf     0.048      42.8
##
## Results are averaged over the levels of: crop
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```r
trt = emmeans(m_flume, pairwise ~ Treatment,
                    type = "response",
                    lmer.df = "asymptotic")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
confint(trt)
```

```
## $emmeans
##  Treatment response     SE  df asymp.LCL asymp.UCL
##  control      0.952 0.6458 Inf    0.2499     3.586
##  strips       0.117 0.0828 Inf    0.0274     0.455
##
## Results are averaged over the levels of: crop, Year
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
## Intervals are back-transformed from the log(mu + 0.005) scale
##
## $contrasts
##  contrast         ratio   SE  df asymp.LCL asymp.UCL
##  control / strips   7.8 7.44 Inf       1.2      50.6
##
## Results are averaged over the levels of: crop, Year
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```r
year = emmeans(m_flume, ~ Year,
                    type = "response",
                    lmer.df = "asymptotic")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
confint(year)
```

```
##  Year response     SE  df asymp.LCL asymp.UCL
##  2016   0.1455 0.0916 Inf    0.0405     0.491
##  2017   0.5731 0.4373 Inf    0.1261     2.540
##  2018   0.4773 0.2508 Inf    0.1690     1.331
##  2019   1.2171 0.6783 Inf    0.4067     3.622
##  2020   0.0867 0.0797 Inf    0.0115     0.497
##
## Results are averaged over the levels of: Treatment, crop
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
## Intervals are back-transformed from the log(mu + 0.005) scale
```

```r
trt_ppt = emmeans(m_flume, pairwise ~ Treatment|ln_ppt,
                    at=list(ln_ppt=c(-4,-3.5,-3,-2)),
                    type = "response",
                     lmer.df = "asymptotic")

confint(trt_ppt)$contrasts ## exp. the values
```

```
## ln_ppt = -4:
##  contrast         ratio    SE  df asymp.LCL asymp.UCL
##  control / strips  4.60  4.69 Inf     0.625      33.9
##
## ln_ppt = -3.5:
##  contrast         ratio    SE  df asymp.LCL asymp.UCL
##  control / strips  6.30  6.07 Inf     0.953      41.6
##
## ln_ppt = -3:
##  contrast         ratio    SE  df asymp.LCL asymp.UCL
##  control / strips  8.62  8.26 Inf     1.320      56.3
##
## ln_ppt = -2:
##  contrast         ratio    SE  df asymp.LCL asymp.UCL
##  control / strips 16.16 17.62 Inf     1.908     136.9
##
## Results are averaged over the levels of: crop, Year
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```
crop      = emmeans(m_flume, pairwise ~ Treatment|crop,
                     type = "response",
                     lmer.df = "asymptotic")

confint(crop)$contrasts
```

```
## crop = corn:
##  contrast         ratio    SE  df asymp.LCL asymp.UCL
##  control / strips 20.40 20.59 Inf     2.821     147.5
##
## crop = soybean:
##  contrast         ratio    SE  df asymp.LCL asymp.UCL
##  control / strips  2.98  3.33 Inf     0.334      26.6
##
## Results are averaged over the levels of: Year
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

Check assumptions

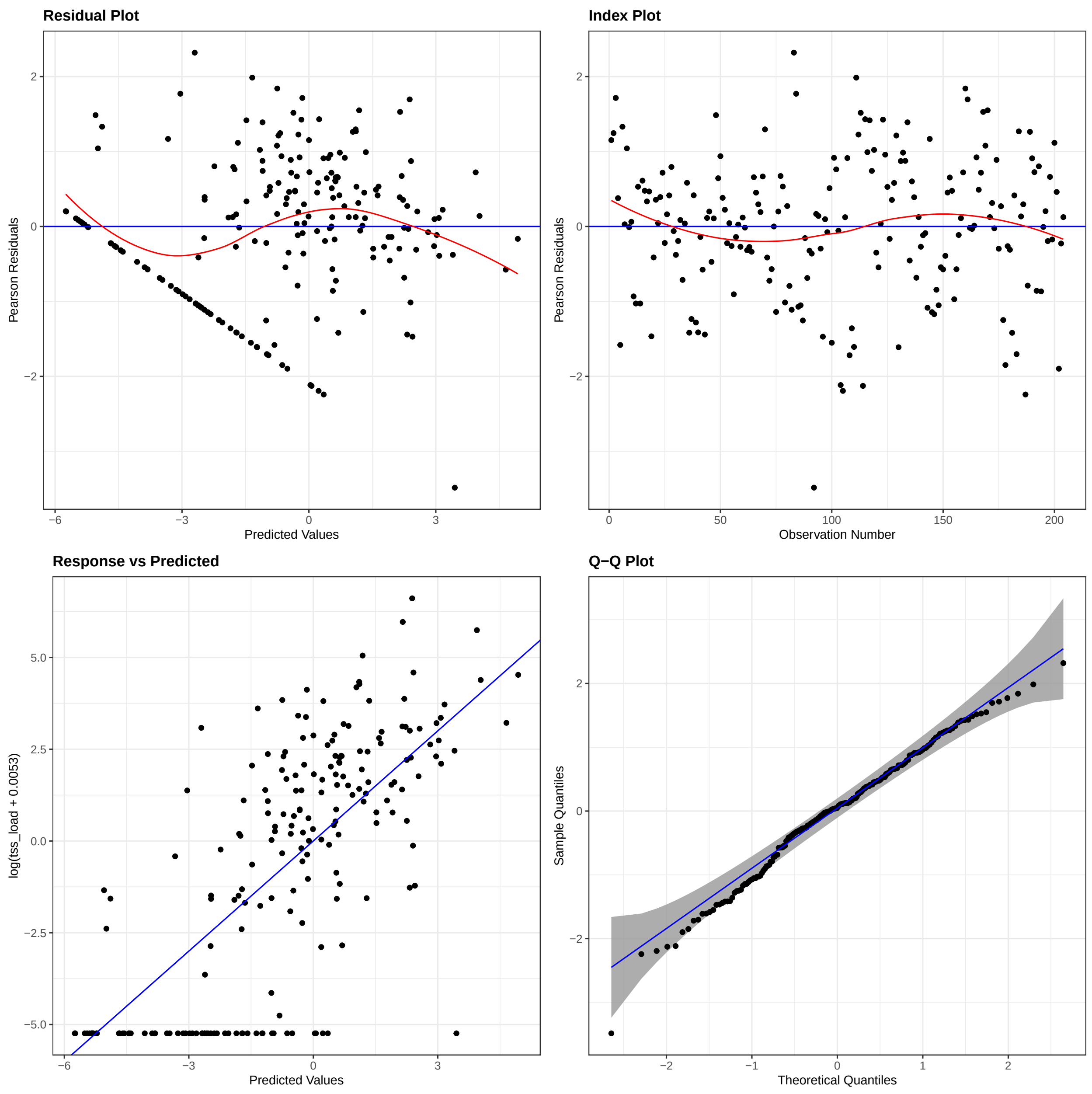
There are two possible models:

- m_flume: full model design, design-based analysis
- m_flume_model: model design selected based on backward step selection

Full model design

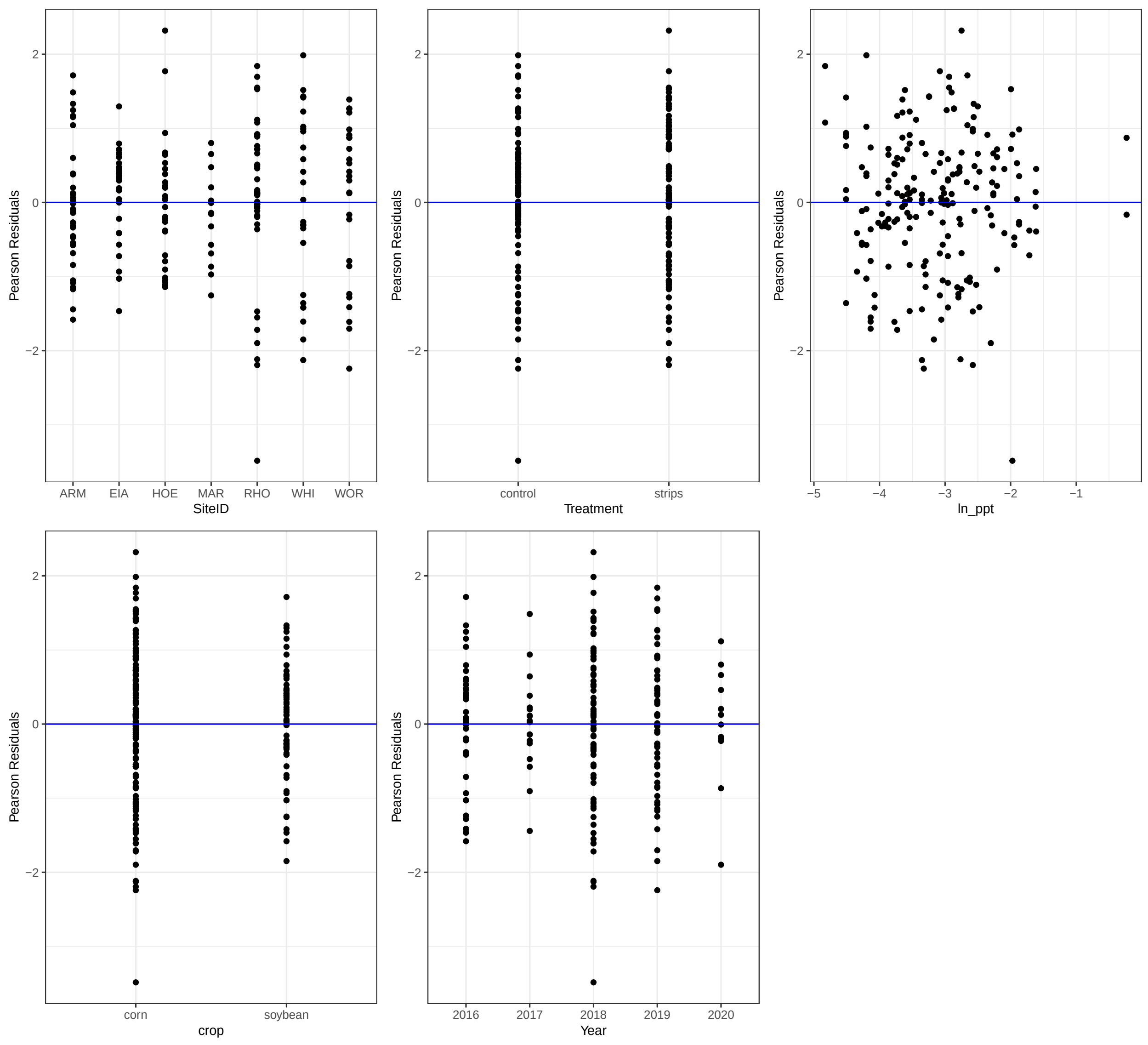
```
resid_panel(m_flume,
            plots = c("resid","index","yvp","qq"),
            smoother = TRUE, qqbands = TRUE)
```

```
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
resid_xpanel(m_flume)
```

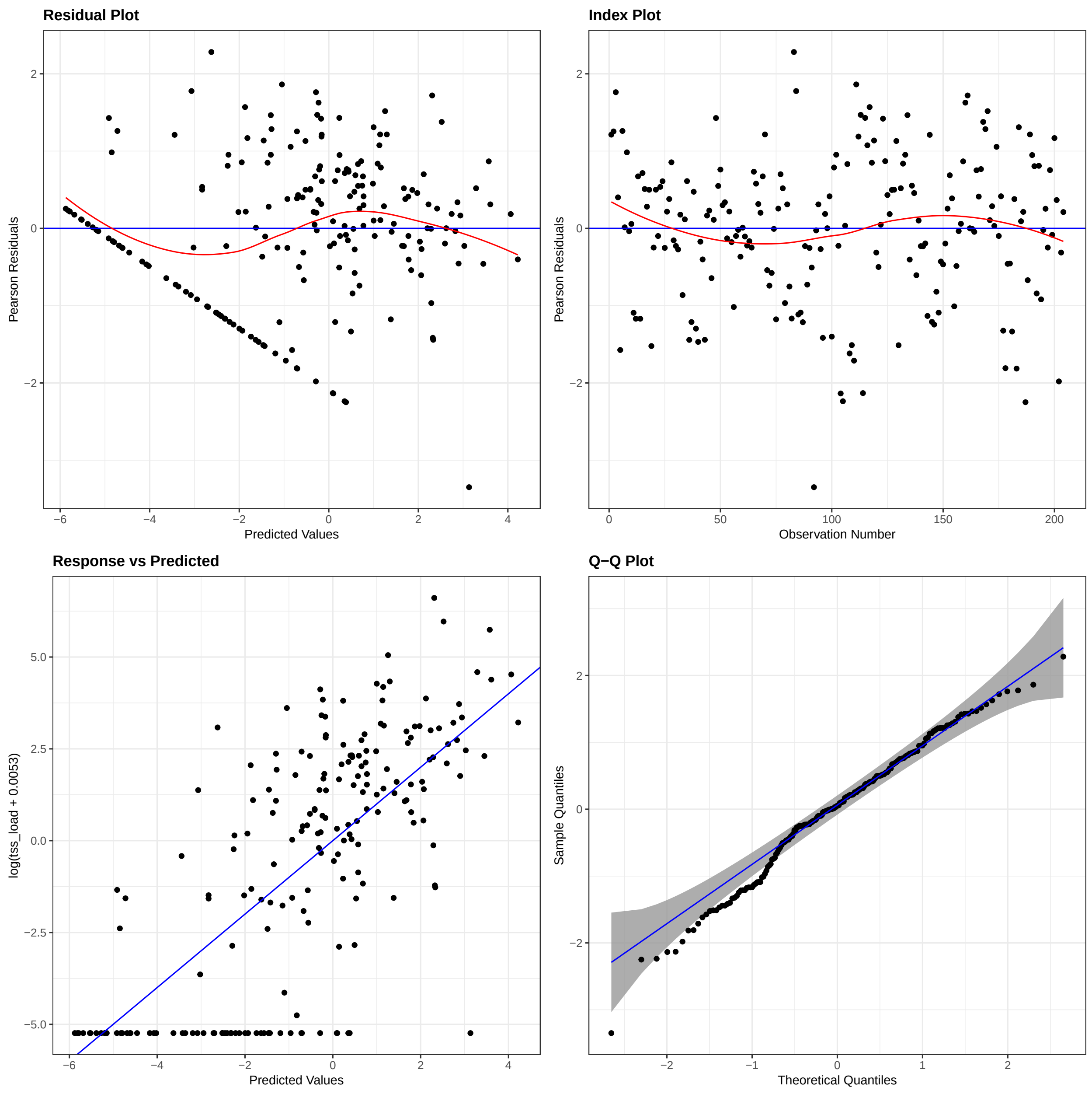
Plots of Residuals vs Predictor Variables



Selected model design

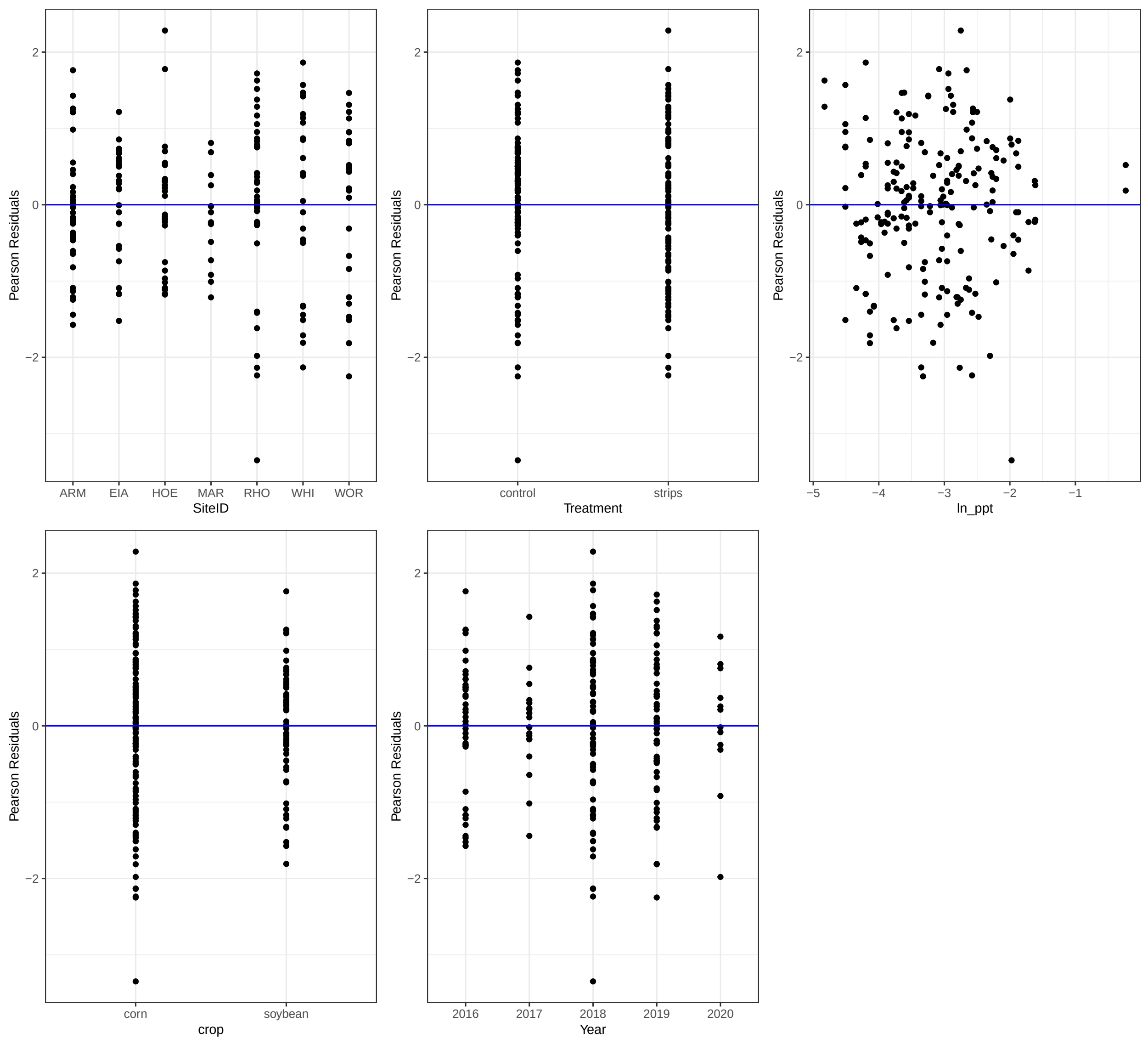
```r
resid_panel(m_flume_model,
            plots = c("resid","index","yvp","qq"),
            smoother = TRUE, qqbands = TRUE)
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



```
resid_xpanel(m_flume_model)
```

Plots of Residuals vs Predictor Variables


Pearson Residuals
ARM
EIA
HOE
MAR
RHO
WHI
WOR
SiteID
control
strips
Treatment
ln_ppt
corn
soybean
crop
Year