

Supplementary code to reproduce the numerical results in Di Caterina and Kosmidis (2017)

Ioannis Kosmidis, Claudia Di Caterina

14 July 2018

Workspace preparation

This page provides R (R Core Team 2017) code to reproduce the results in the manuscript ‘Location-adjusted Wald statistic for scalar parameters’ (Di Caterina and Kosmidis 2017).

This script assumes that the working directory is set to `supplementary_1710-11217`. If you have placed the contents of `supplementary_1710-11217` elsewhere, then change `path` appropriately.

```
path <- "."
code_path <- paste(path, "code", sep = "/")
results_path <- paste(path, "results", sep = "/")
```

First, make sure that you have the latest version of the **waldi** R package installed.

```
waldi_version <- try(packageVersion("waldi"), silent = TRUE)
if (inherits(waldi_version, "try-error")) {
  devtools::install_github("ikosmidis/waldi")
}
```

The following code chunk loads the required packages

```
library("waldi")
library("oro.nifti")
library("boot")
library("plyr")
library("plotrix")
library("dplyr")
library("survival")
library("cond")
library("lmtest")
library("betareg")
library("enrichwith")
library("brglm2")
library("ggplot2")
library("gridExtra")
library("colorspace")
```

Pre-saved R image files

Some of the code-chunks below load objects from the pre-saved R image files in the results directory. These image files are the outputs of the script `babies_simulation.R`, `brockwell_gordon_simulation.R`, `clotting_simulation.R`, `dyslexia_simulation.R`.

Table 1

```
data("ReadingSkills", package = "betareg")

## maximum likelihood estimates and corresponding 95% Wald confidence intervals
rs_beta_ml <- betareg(accuracy ~ dyslexia * iq | dyslexia + iq,
                     data = ReadingSkills, type = "ML", hessian = FALSE)
rs_summary_ml <- coef(summary(rs_beta_ml))
rs_ml_estimates <- do.call("rbind", lapply(rs_summary_ml,
                                           function(z) z[, c("Estimate", "Std. Error")]))
rs_ml_cis <- confint(rs_beta_ml)

## bias corrected fit and corresponding 95% Wald confidence intervals
rs_beta_br <- update(rs_beta_ml, type = "BR")
rs_summary_br <- coef(summary(rs_beta_br))
rs_br_estimates <- do.call("rbind", lapply(rs_summary_br,
                                           function(z) z[, c("Estimate", "Std. Error")]))
rs_br_cis <- confint(rs_beta_br)

round(cbind(rs_ml_estimates, rs_br_estimates, rs_ml_cis, rs_br_cis), 3)
#           Estimate Std. Error Estimate Std. Error 2.5 % 97.5 % 2.5 % 97.5 %
# (Intercept)   1.123    0.143    1.114    0.148  0.843  1.403  0.824  1.405
# dyslexia      -0.742    0.143   -0.734    0.148 -1.021 -0.462 -1.024 -0.444
# iq            0.486    0.133    0.441    0.141  0.225  0.747  0.165  0.717
# dyslexia:iq   -0.581    0.133   -0.532    0.140 -0.841 -0.321 -0.807 -0.257
# (Intercept)   3.304    0.223    3.092    0.225  2.868  3.741  2.652  3.533
# dyslexia      1.747    0.262    1.654    0.264  1.232  2.261  1.138  2.171
# iq            1.229    0.267    1.048    0.271  0.705  1.753  0.518  1.578
```

Table 2

dyslexia_simulation.rda below is the output of dyslexia_simulation.R in ./code, which replicates the simulation study described in Example 1.1 of Di Caterina and Kosmidis (2017)

```
load(paste(results_path, "dyslexia_simulation.rda", sep = "/"))

## typeI error
typeI <- ddpI(res, ~ statistic + parameter, function(x) {
  levels <- c(0.1, 1, 2.5, 5, 10)/100
  p_value_2sided <- 2 * pnorm(-abs(x$value))
  p_value_left <- pnorm(x$value)
  p_value_right <- 1 - pnorm(x$value)
  rate_2sided <- sapply(levels, function(alpha) mean(p_value_2sided < alpha))
  rate_left <- sapply(levels, function(alpha) mean(p_value_left < alpha))
  rate_right <- sapply(levels, function(alpha) mean(p_value_right < alpha))
  out <- data.frame(
    test = rep(c("2sided", "left", "right"), each = length(levels)),
    typeI = c(rate_2sided, rate_left, rate_right),
    level = rep(levels, times = 3))
  out
})
```

```
## compute coverage probabilities
rs_coverage <- typeI %>%
  filter((statistic %in% c("mle", "br")) &
    level %in% c(0.1, 0.05, 0.01) &
    test == "2sided" &
    parameter %in% c(2, 3, 4, 6, 7)) %>%
  select(-test) %>%
  mutate(coverage = round(100 * (1 - typeI), 1)) %>%
  mutate(level = 100 * (1 - level)) %>%
  select(-typeI) %>%
  arrange(level) %>%
  reshape(idvar = c("level", "parameter"), v.names = "coverage",
    timevar = "statistic",
    direction = "wide")
rs_coverage
#   parameter level coverage.br coverage.mle
# 1           2    90      88.1      86.9
# 2           3    90      87.2      84.8
# 3           4    90      87.3      85.0
# 4           6    90      83.8      82.4
# 5           7    90      82.7      79.1
# 11          2    95      93.4      92.4
# 12          3    95      92.9      91.0
# 13          4    95      92.9      91.2
# 14          6    95      90.2      89.1
# 15          7    95      89.2      86.0
# 21          2    99      98.2      97.7
# 22          3    99      98.0      97.1
# 23          4    99      98.0      97.2
# 24          6    99      96.7      96.1
# 25          7    99      96.1      94.4
```

Figure 1

```
rs_cor_ml_cis <- waldci_confint(rs_beta_ml, level = 0.95, adjust = TRUE)
# Warning in setup_parallel(): No parallel backend registered
interpolation <- waldci_confint(rs_beta_ml, level = 0.95,
  which = rownames(rs_cor_ml_cis),
  adjust = TRUE,
  return_values = TRUE)
# Warning in setup_parallel(): No parallel backend registered

intervals <- data.frame(low = rs_cor_ml_cis[, 1],
  upp = rs_cor_ml_cis[, 2],
  parameter = rownames(rs_cor_ml_cis))

interpolation <- interpolation %>%
  filter(!(parameter %in% c("(Intercept)", "(phi)_(Intercept)"))) %>%
  mutate(parameter = recode(parameter,
    "dyslexia" = "beta[2]",
    "iq" = "beta[3]",
```

```

      "dyslexia:iq" = "beta[4]",
      "(phi)_dyslexia" = "gamma[2]",
      "(phi)_iq" = "gamma[3]")

intervals <- intervals %>%
  filter(!(parameter %in% c("(Intercept)", "(phi)_(Intercept)"))) %>%
  mutate(parameter = recode(parameter,
    "dyslexia" = "beta[2]",
    "iq" = "beta[3]",
    "dyslexia:iq" = "beta[4]",
    "(phi)_dyslexia" = "gamma[2]",
    "(phi)_iq" = "gamma[3]"))

ggplot(interpolation) +
  geom_point(aes(x = grid, y = value)) +
  geom_line(aes(x = grid, y = value), col = "grey") +
  geom_hline(aes(yintercept = qnorm(0.975)), col = "grey", lty = 3) +
  geom_hline(aes(yintercept = qnorm(0.025)), col = "grey", lty = 3) +
  geom_vline(data = intervals, aes(xintercept = low), col = "grey", lty = 2) +
  geom_vline(data = intervals, aes(xintercept = upp), col = "grey", lty = 2) +
  facet_grid(~ parameter, scale = "free_x", labeller = "label_parsed") +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 7)) +
  labs(x = "parameter value", y = "statistic")

```

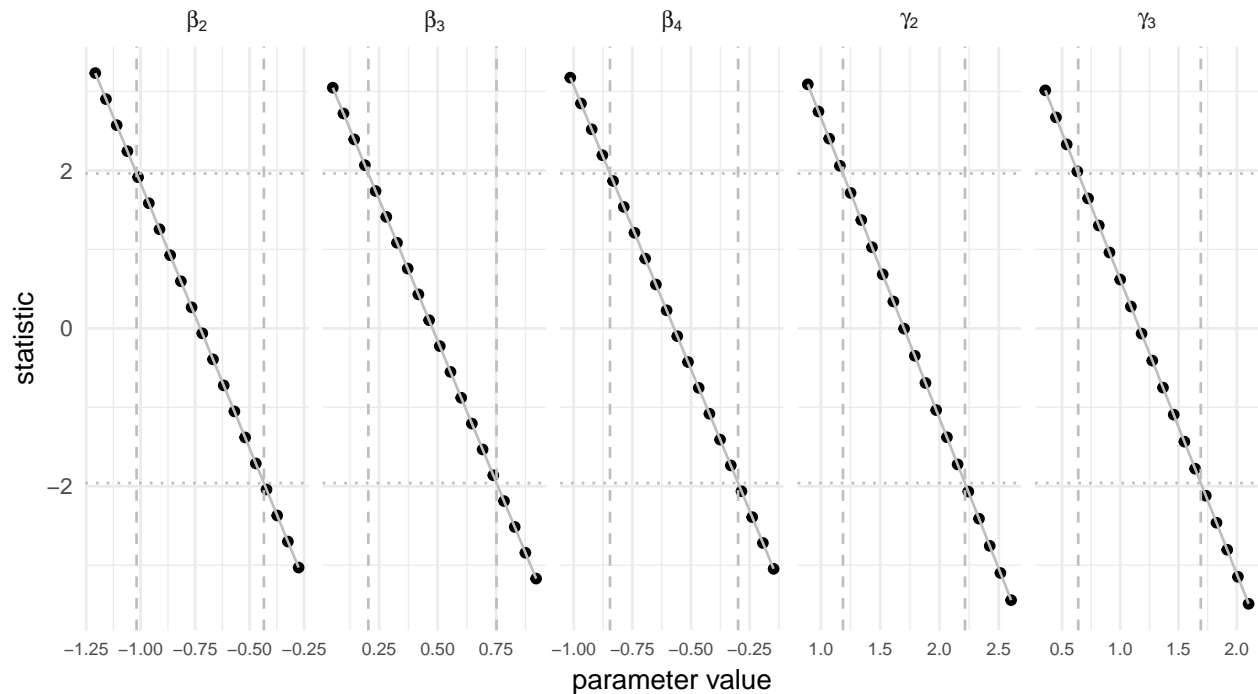


Table 3

```

rs_cor_ml_cis <- waldi_confint(rs_beta_ml, level = 0.95, adjust = TRUE)
# Warning in setup_parallel(): No parallel backend registered

```

```

rs_cor_br_cis <- waldci_confint(rs_beta_br, level = 0.95, adjust = TRUE)
# Warning in setup_parallel(): No parallel backend registered

rs_coverage <- typeI %>%
  filter((statistic %in% c("mle_cor", "br_cor")) &
    level %in% c(0.1, 0.05, 0.01) &
    test == "2sided" &
    parameter %in% c(2, 3, 4, 6, 7)) %>%
  select(-test) %>%
  mutate(coverage = round(100 * (1 - typeI), 1)) %>%
  mutate(level = 100 * (1 - level)) %>%
  select(-typeI) %>%
  arrange(level) %>%
  reshape(idvar = c("level", "parameter"), v.names = "coverage",
    timevar = "statistic",
    direction = "wide")

round(cbind(rs_cor_ml_cis, rs_cor_br_cis), 3)
#               2.5 % 97.5 %   2.5 % 97.5 %
# (Intercept)    0.816  1.400  0.827  1.411
# dyslexia       -1.019 -0.435 -1.031 -0.446
# iq             0.204  0.752  0.165  0.719
# dyslexia:iq    -0.845 -0.299 -0.809 -0.257
# (phi)_ (Intercept) 2.689  3.564  2.652  3.532
# (phi)_dyslexia    1.186  2.214  1.134  2.169
# (phi)_iq         0.639  1.691  0.513  1.574
rs_coverage
#   parameter level coverage.br_cor coverage.mle_cor
# 1         2    90           88.3           88.5
# 2         3    90           87.3           87.1
# 3         4    90           87.5           87.2
# 4         6    90           83.9           83.5
# 5         7    90           82.7           81.8
# 11        2    95           93.5           93.7
# 12        3    95           93.0           92.8
# 13        4    95           93.0           92.8
# 14        6    95           90.3           90.0
# 15        7    95           89.2           88.6
# 21        2    99           98.3           98.4
# 22        3    99           98.0           98.0
# 23        4    99           98.0           98.0
# 24        6    99           96.8           96.6
# 25        7    99           96.2           95.7

```

Figure 2

```

source(paste0(code_path, "/", "logodds_functions.R"))

## Distribution of the statistic against normal
settings <- expand.grid(m = c(8, 16, 32), theta0 = c(-2, -1, 0))
plot_data <- NULL

```

```

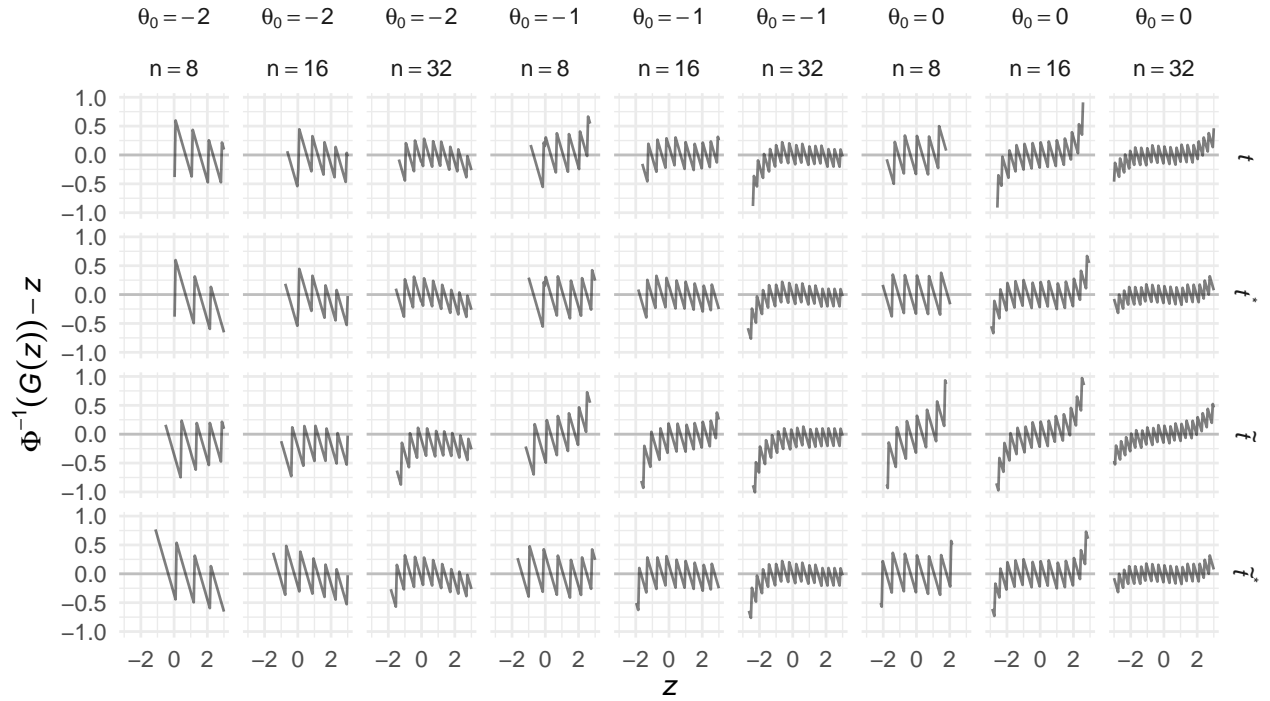
for (j in seq.int(nrow(settings))) {
  setting <- settings[j, ]
  z <- seq(-3, 3, length = 100)
  dat <- t(sapply(z, dist_function, n = setting$m, theta0 = setting$theta0))
  dd <- stack(as.data.frame(dat))
  dd$z <- z
  names(dd) <- c("prob", "method", "z")
  dd$theta0 <- setting$theta0
  dd$m <- setting$m
  plot_data <- rbind(plot_data, dd)
}

plot_data$theta0 <- paste0("theta[0] == ", plot_data$theta0)
plot_data$theta0 <- factor(plot_data$theta0, levels = unique(plot_data$theta0),
                           ordered = TRUE)
plot_data$m <- paste0("n == ", plot_data$m)
plot_data$m <- factor(plot_data$m, levels = unique(plot_data$m), ordered = TRUE)
plot_data$method <- factor(plot_data$method, levels = c("ml", "a_ml", "br", "a_br"),
                           ordered = TRUE)
plot_data$method <- recode(plot_data$method,
                           "ml" = "italic(t)",
                           "a_ml" = "italic(t)^{'*'}",
                           "br" = "tilde(italic(t))",
                           "a_br" = "tilde(italic(t))^{ '* }")

ggplot(plot_data) +
  geom_abline(aes(intercept = 0, slope = 0), col = "grey") +
  geom_line(aes(z, qnorm(prob) - z), alpha = 0.5) +
  facet_grid(method ~ theta0 + m, label = label_parsed) +
  theme_minimal() +
  labs(y = expression(paste(Phi^list(-1), (italic(G)(italic(z))) - italic(z))),
       x = expression(italic(z))) +
  theme(text=element_text(size = 11))
# Warning in qnorm(prob): NaNs produced

# Warning in qnorm(prob): NaNs produced
# Warning: Removed 50 rows containing missing values (geom_path).

```



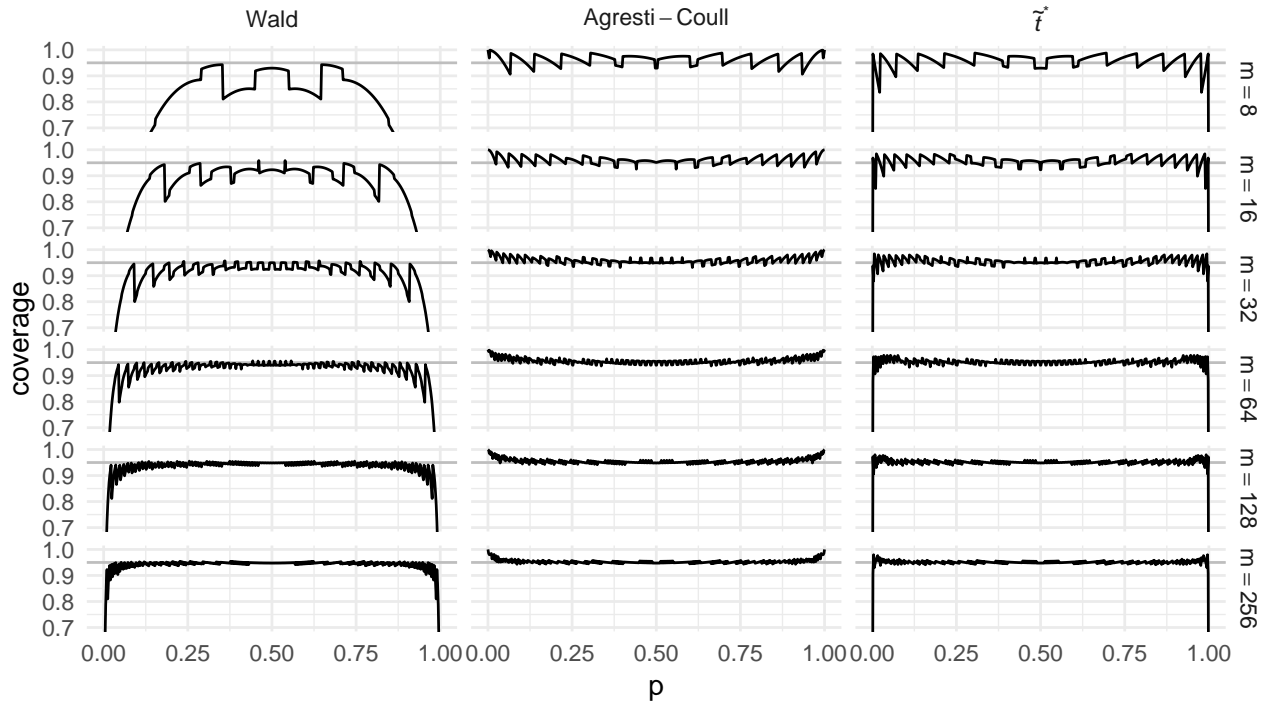
Coverage and length of confidence intervals for a binomial proportion

This section provides evidence for the stated coverage and expected length properties of confidence intervals for a binomial proportion in Section 7 of the main text. The code chunk below computes and visualised the coverage and expected length of the 95% confidence intervals $\bar{y} \pm z_{0.975} \sqrt{\bar{y}(1-\bar{y})/n}$ (Wald), $\tilde{p} \pm z_{0.975} \sqrt{\tilde{p}(1-\tilde{p})/(n+4)}$, where $\tilde{p} = (\sum y_i + 2)/(n+4)$ (Agresti-Coull; Agresti and Coull (1998) and Agresti and Caffo (2000)), and the intervals based on the transformation of the endpoints of the confidence intervals for the log-odds based on t^* .

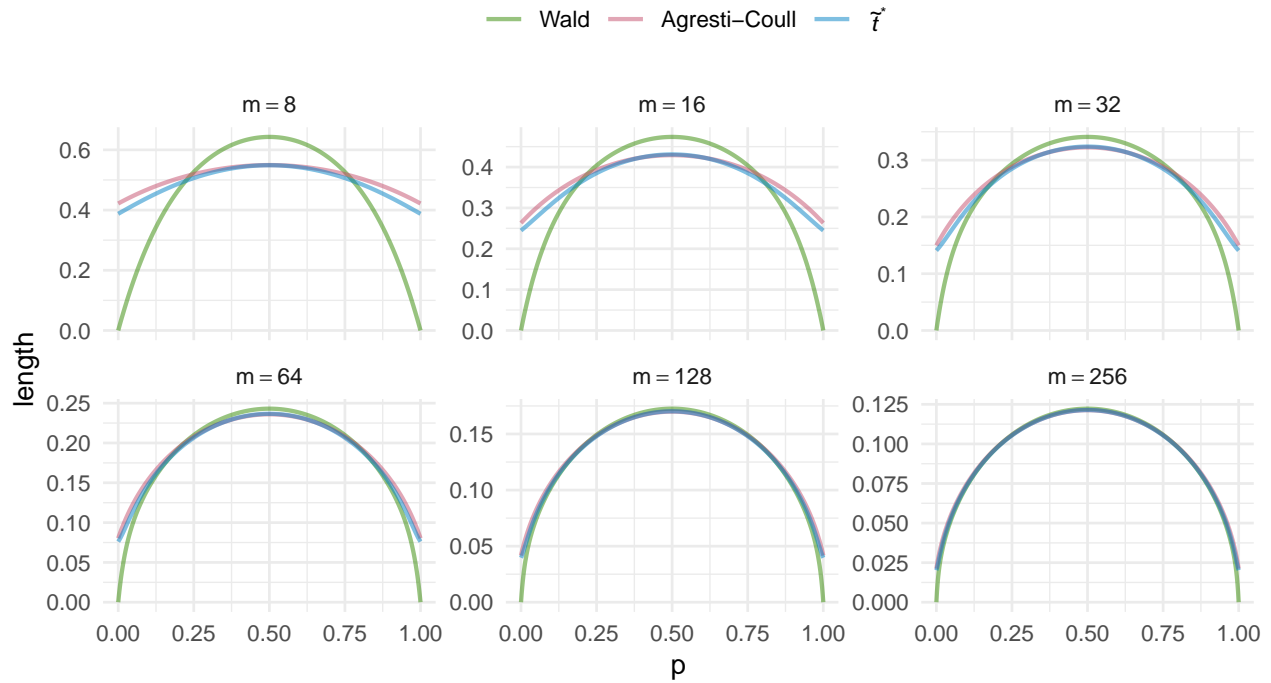
```
probs <- seq(1e-08, 1 - 1e-08, length = 500)
df <- ddply(data.frame(m = c(8, 16, 32, 64, 128, 256)), ~ m, function(x) {
  m <- x$m
  cis <- compute_cis(m, level = 0.95)
  cc <- lapply(probs, function(pp) cover_ci_prop(n = m, p = pp, level = 0.95, cis = cis))
  do.call("rbind", cc)
})

df$m <- factor(paste("m ==", df$m), levels = paste("m ==", sort(unique(df$m))),
              ordered = TRUE)
df$method <- factor(df$method, levels = c("wald", "ac", "a_br", "ml", "a_ml", "br"),
                  ordered = TRUE)
df$method <- recode(df$method,
                  "wald" = "Wald",
                  "ml" = "italic(t)[trans]",
                  "a_ml" = "italic(t)^[trans]",
                  "br" = "tilde(italic(t))[trans]",
                  "a_br" = "tilde(italic(t))^[*]",
                  "ac" = "Agresti-Coull")
```

```
## coverage
ggplot(df %>% filter(method %in% c("Wald", "Agresti-Coull", "tilde(italic(t))^{ '*' }"))) +
  geom_hline(aes(yintercept = 0.95), col = "grey") +
  geom_line(aes(x = p, y = coverage)) +
  facet_grid(m ~ method, label = label_parsed) +
  coord_cartesian(ylim = c(0.7, 1)) +
  theme_minimal()
```



```
## expected length
ggplot(df %>% filter(method %in% c("Wald", "Agresti-Coull", "tilde(italic(t))^{ '*' }"))) +
  geom_line(aes(x = p, y = length, col = method, alpha = 0.5, size = 0.8)) +
  facet_wrap(~ m, label = label_parsed, scales = "free_y", ncol = 3) +
  scale_colour_manual(values = c("#328900", "#C54E6D", "#0080C5"),
    name = "",
    labels = c("Wald",
      "Agresti-Coull",
      bquote(tilde(italic(t))^{ '*' }))) +
  theme_minimal() +
  theme(legend.position = "top")
```

Hauck and Donner effect

```
sapply(28:32, t_ml, n = 32, theta0 = 0)
# [1] 3.640465 3.740749 3.708150 3.379905 0.000000
sapply(28:32, t_adjusted_ml, n = 32, theta0 = 0)
# [1] 3.770481 3.912737 3.955360 3.816022 0.000000
sapply(28:32, t_br, n = 32, theta0 = 0)
# [1] 3.583279 3.712935 3.744298 3.587411 2.884566
sapply(28:32, t_adjusted_br, n = 32, theta0 = 0)
# [1] 3.763721 3.902155 3.935838 3.762302 2.921237
```

Table 4

```
## The clotting data set
clotting <- data.frame(
  conc = c(118,58,42,35,27,25,21,19,18,69,35,26,21,18,16,13,12,12),
  u = c(5,10,15,20,30,40,60,80,100, 5,10,15,20,30,40,60,80,100),
  lot = factor(c(rep(1, 9), rep(2, 9))))

## The maximum likelihood fit of the gamma regression model
clotting_ml <- glm(conc ~ log(u)*lot, data = clotting, family = Gamma(link = "log"))

## Maximum likelihood estimates and Wald statistics using maximum likelihood estimator
## of the dispersion parameter
dispersion_ml <- MASS::gamma.dispersion(clotting_ml)
clotting_summary_ml <- summary(clotting_ml, dispersion = dispersion_ml)
clotting_ml_estimates <- coef(clotting_summary_ml)[, c("Estimate", "z value")]
```

```

## Reduced-bias estimates and Wald statistics
clotting_summary_rb <- summary(update(clotting_ml, method = "brglmFit"))

## Maximum likelihood estimates and Wald statistics using the moment-based estimator
## of the dispersion parameter
clotting_summary_mom <- summary(clotting_ml)
dispersion_mom <- clotting_summary_mom$dispersion
clotting_mom_estimates <- coef(clotting_summary_mom)[, c("Estimate", "t value")]

## Location-adjusted Wald statistic
clotting_waldi <- waldi(clotting_ml, null = 0, adjust = TRUE)

round(cbind(c(clotting_ml_estimates[, 1], dispersion_ml, dispersion_mom),
             c(clotting_ml_estimates[, 2], NA, NA),
             c(clotting_mom_estimates[, 2], NA, NA),
             c(clotting_waldi, NA, NA)), 3)
#           [,1]    [,2]    [,3]    [,4]
# (Intercept)  5.503  34.124  29.282  28.953
# log(u)       -0.602 -12.842 -11.020 -10.896
# lot2         -0.584  -2.563  -2.199  -2.173
# log(u):lot2   0.034   0.520   0.446   0.441
#              0.017    NA      NA      NA
#              0.024    NA      NA      NA

```

Figure 3 including rejection probabilities based on t_j^* and the Wald statistic using $\tilde{\phi}$

clotting_simulation.rda below is the output of clotting_simulation.R in ./code, which replicates the simulation study described in Section 8.3 of Di Caterina and Kosmidis (2017).

```

load(paste(results_path, "clotting_simulation.rda", sep = "/"))

## Compute type I error rates
typeI <- ddply(res, ~ statistic + parameter, function(x) {
  ## empirical <- pnorm(quantile(x$value, c(0, 1, 2.5, 5, 1)/100))
  levels <- c(0.1, 1, 2.5, 5)/100
  p_value_2sided <- 2 * pnorm(-abs(x$value))
  p_value_left <- pnorm(x$value)
  p_value_right <- 1 - pnorm(x$value)
  rate_2sided <- sapply(levels, function(alpha) mean(p_value_2sided < alpha))
  rate_left <- sapply(levels, function(alpha) mean(p_value_left < alpha))
  rate_right <- sapply(levels, function(alpha) mean(p_value_right < alpha))
  out <- data.frame(
    test = rep(c("2sided", "left", "right"), each = length(levels)),
    typeI = c(rate_2sided, rate_left, rate_right),
    level = rep(levels, times = 3))
  out
})

typeI <- typeI %>%
  filter(test != "right") %>%

```

```

mutate(test = recode(test,
  "2sided" = "beta[italic(j)] != beta[paste(italic(j), 0)]",
  "left" = "beta[italic(j)] < beta[paste(italic(j), 0)]",
  "right" = "beta[italic(j)] > beta[paste(italic(j), 0)]",
  level_chr = paste(level*100, "~symbol('\045')"),
  upper = typeI - qnorm(1 - 0.01/2)*sqrt(typeI*(1-typeI)/nsimu),
  lower = typeI + qnorm(1 - 0.01/2)*sqrt(typeI*(1-typeI)/nsimu))

## Figure 2 in the manuscript
ggplot(typeI %>% filter(parameter != 1)) +
  geom_point(aes(parameter, typeI, pch = statistic), alpha = 0.7) +
  geom_hline(aes(yintercept = level), col = "grey", lty = 2) +
  facet_grid(test ~ level_chr, labeller = label_parsed, scales = "free") +
  scale_x_continuous(name = element_blank(),
    breaks = c(2, 3, 4),
    limits = c(1.8, 4.2),
    labels = c(
      expression(beta[2]),
      expression(beta[3]),
      expression(beta[4])) +
  scale_y_continuous(name = expression(paste("Empirical rejection probability (",
    symbol('\045'), ")")),
    labels = function (x) {
      if (length(x) == 0)
        return(character())
      x <- round_any(x, scales::precision(x)/100)
      scales::comma(x * 100)
    }) +
  theme_bw() +
  theme(legend.position = "top",
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.minor.x = element_blank(),
    strip.background = element_blank())

```

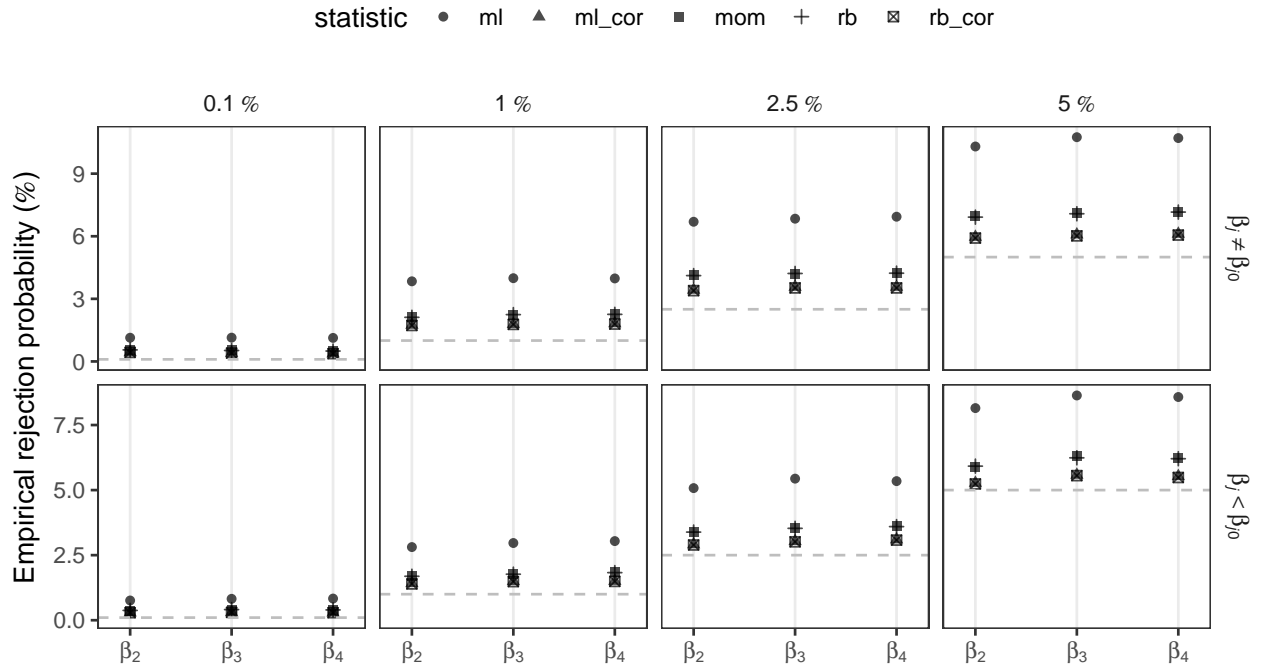


Table 5

```
data("babies", package = "cond")

## clogit understands only 0-1 so expand
babies_expand <- ddply(babies, ~ lull + day, function(z) {
  data.frame(y = rep(c(0, 1), c(z$r2, z$r1)))
})

## Maximum likelihood fit
babies_ml <- glm(formula = y ~ day + lull - 1,
  family = binomial, data = babies_expand)

babies_rb <- update(babies_ml, method = "brglmFit")

## Maximum conditional likelihood fit
babies_cond <- clogit(y ~ strata(day) + lull, data = babies_expand)

ml <- coef(summary(babies_ml))["lullyes", ]
rb <- coef(summary(babies_rb))["lullyes", ]
mcl <- coef(summary(babies_cond))["lullyes", ]
r <- lrtest(update(babies_ml, . ~ . - lull),
  babies_ml)
rc <- summary(babies_cond)$logtest[1]
scorec <- summary(babies_cond)$sctest[1]
out1 <- c(
  ml = unname(ml["Estimate"]),
  rb = unname(rb["Estimate"]),
  mcl = unname(mcl["coef"]),
  wald_ml = unname(ml["z value"]),
```

```

wald_mcl = unname(mcl["z"]),
wald_rb = unname(rb["z value"]),
r = unname(sign(ml["Estimate"]) * sqrt(r$Chisq[2])),
rc = unname(sign(mcl["coef"]) * sqrt(rc)),
wald_ml_adjusted = unname(waldi(babies_ml, which = 19)),
wald_rb_adjusted = unname(waldi(babies_rb, which = 19))

out2 <- c(
  ml_se = unname(ml["Std. Error"]),
  rb_se = unname(rb["Std. Error"]),
  mcl_se = unname(mcl["se(coef)"]),
  ml_p = ml["Pr(>|z|)"],
  mcl_p = mcl["Pr(>|z|)"],
  rb_p = rb["Pr(>|z|)"],
  r_p = 2 * pnorm(-abs(out1["r"])),
  rc_p = 2 * pnorm(-abs(out1["rc"])),
  cor_ml_p = 2 * pnorm(-abs(out1["wald_ml_adjusted"])),
  cor_rb_p = 2 * pnorm(-abs(out1["wald_rb_adjusted"])))

round(matrix(c(out1, out2), ncol = 10, byrow = TRUE,
  dimnames = list(NULL,
    c("mle", "rb", "mcl", "wald_ml", "wald_mlc",
      "wald_rb", "r", "rc", "wald_ml_adjusted",
      "wald_rb_adjusted"))), 4)

#           mle      rb    mcl wald_ml wald_mlc wald_rb      r      rc
# [1,] 1.4324 1.1562 1.2561 1.9511 1.8307 1.7362 2.1596 2.0214
# [2,] 0.7341 0.6659 0.6861 0.0510 0.0671 0.0825 0.0308 0.0432
#           wald_ml_adjusted wald_rb_adjusted
# [1,]                1.9257                1.9064
# [2,]                0.0541                0.0566

```

Figure 4

babies_simulation.rda below is the output of babies_simulation.R in ./code, which replicates the simulation study described in Section 8.4 of Di Caterina and Kosmidis (2017)

```

load(paste(results_path, "babies_simulation.rda", sep = "/"))

## The bootstrap p-value for the babies data is
set.seed(123)
babies_bootstrap(babies_ml, R = 1000)$conv
# [1] 0.0230001

## Compute pvalues from the various statistics account for the existence of bootstrap
## p-values
pval <- ddply(res %>% filter(!infinite & !is.na(value) & type != "summary"),
  ~ name,
  function(data) {
    if (all(data$type == "bootstrap_statistic")) {
      data.frame(sample = pnorm(data$value),
        test = gsub("boot_prep_|boot_conv_", "", data$name))
    }
    else {

```

```

    p2 <- 2 * pnorm(-abs(data$value))
    p1 <- pnorm(data$value)
    pr <- 1 - p1
    data.frame(sample = c(p2, p1, pr),
               test = rep(c("2sided", "left", "right"), each = length(p2))) }
  })

## Get rid of left right 2sided from statistic names
pval <- pval %>% mutate(name = gsub("_left|_right|_2sided", "", name))

pval <- pval %>%
  filter(!(name %in% c("scorec", "boot_prep"))) & test != "right" %>%
  mutate(test = dplyr::recode(test,
                             "2sided" = "gamma != 0",
                             "left" = "gamma < 0",
                             "right" = "gamma > 0"),
         name = factor(name,
                       levels = c("mle", "rbe", "r", "cond", "scorec", "rc",
                                   "boot_conv", "cor", "cor_rb"),
                       ordered = TRUE)) %>%
  mutate(name = factor(name,
                       levels = c("mle", "r", "boot_conv", "rbe",
                                   "cond", "scorec", "rc",
                                   "cor", "cor_rb"),
                       ordered = TRUE)) %>%
  mutate(statistic = dplyr::recode(name,
                                   "mle" = "italic(t)",
                                   "rbe" = "italic(tilde(t))",
                                   "r" = "italic(r)",
                                   "cond" = "italic(t)[c]",
                                   "scorec" = "italic(s)[c]",
                                   "rc" = "italic(r)[c]",
                                   "cor" = "italic(t)~'*'",
                                   "cor_rb" = "tilde(italic(t))~'*'",
                                   "boot_conv" = "italic(boot)"))

## Bin sample
breaks <- (0:20)/20
pval <- pval %>%
  group_by(statistic, test) %>%
  mutate(sample = cut(sample, breaks = breaks, include.lowest = TRUE)) %>%
  group_by(statistic, test, sample)

ggplot(pval) +
  geom_hline(aes(yintercept = 1)) +
  geom_bar(aes(x = sample, y = ..count../2500), fill = "darkgray", alpha = 0.5) +
  facet_grid(test ~ statistic, labeller = label_parsed) +
  theme_bw() +
  scale_x_discrete(breaks = c("[0,0.05]", "(0.25,0.3]", "(0.5,0.55]",
                              "(0.75,0.8]", "(0.95,1]"),
                  labels = c(0, 0.25, 0.5, 0.75, 1)) +
  theme(legend.position = "top",
        panel.grid.major.y = element_blank(),

```

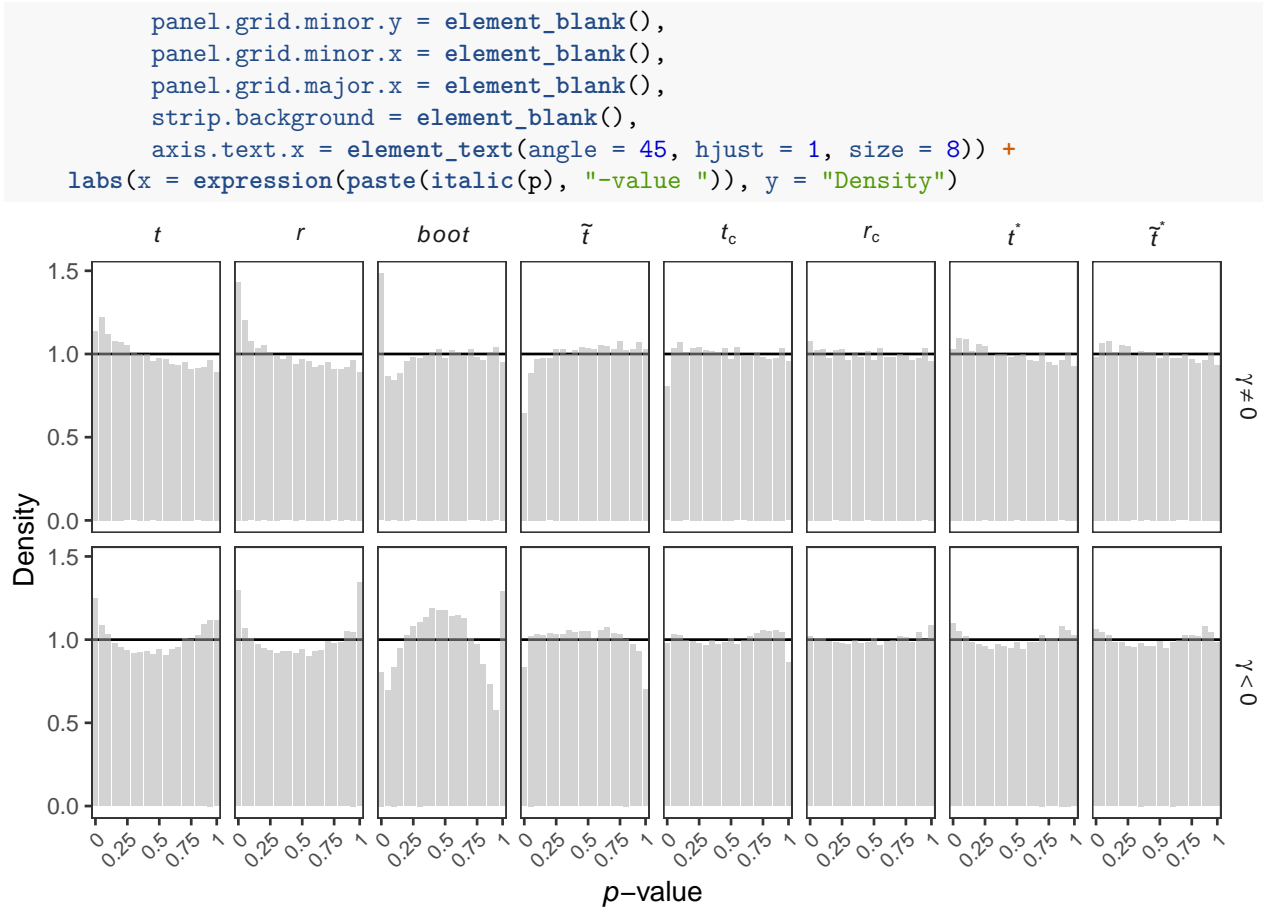


Figure 5

brains_case_study.rda below is the output of brains_case_study.R in ./code, which replicates the simulation study described in Section 9 of Di Caterina and Kosmidis (2017)

```

source(paste0(code_path, "/", "overlay2_nifti.R"))
load(paste(results_path, "brains_case_study.rda", sep = "/"))

## Check how many times LR failed, excluding trivial voxels, and compute probability of
## infinite estimates
fits_mat %>% filter(statistic == "I" & voxel != 1) %>% group_by(parameter) %>%
  summarize(failed = 100 * sum((value == -Inf) * count) / sum(count),
            infinite = 100 * sum(infinite * count) / sum(count))

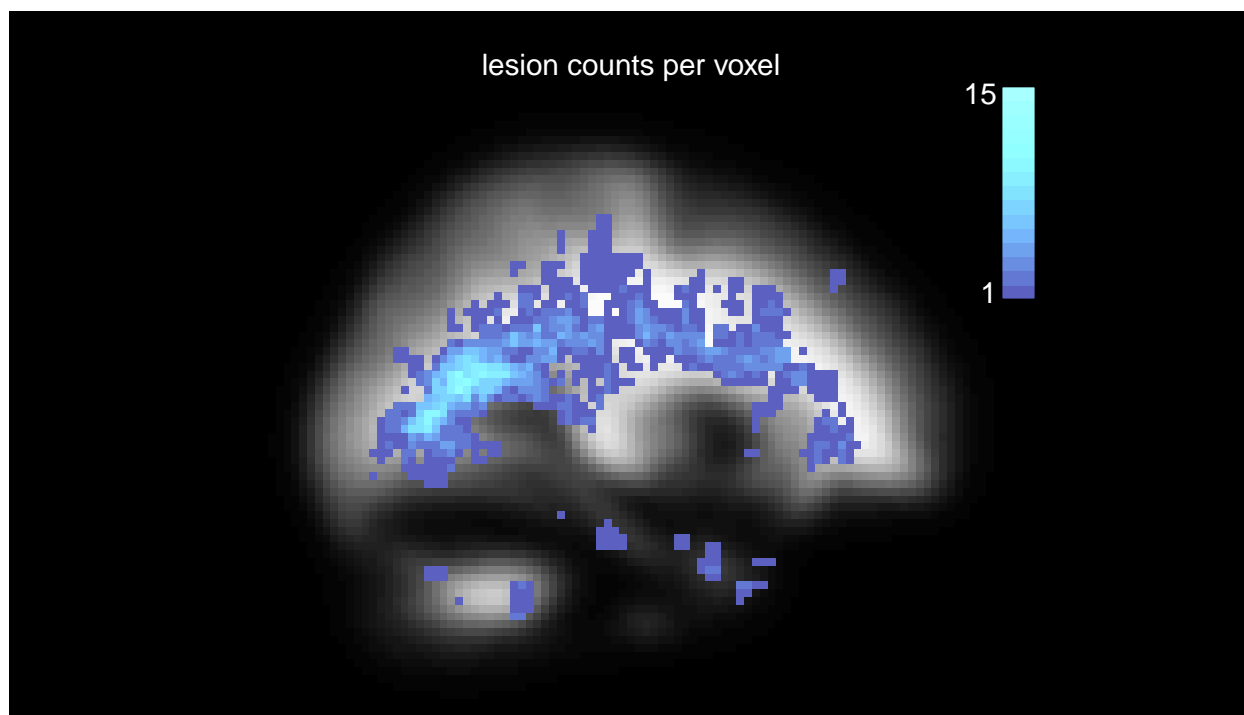
## A tibble: 6 x 3
##   parameter failed infinite
##   <fct>      <dbl>    <dbl>
## 1 age       20.5      63.7
## 2 DD       18.1      63.7
## 3 EDSS     10.3      63.2
## 4 PASAT    16.8      63.6
## 5 sex      22.4      78.3
## 6 type2    19.2      75.5

```

```
## detections
fits_mat %>%
  group_by(parameter, statistic) %>%
  filter(statistic %in% c("z_br", "corz_br")) %>%
  summarize(detections = mean(value < -1 | value > 1) * 100)

# # A tibble: 12 x 3
# # Groups:   parameter [?]
#   parameter statistic detections
#   <fct>      <fct>      <dbl>
# 1 age       corz_br       39.2
# 2 age       z_br         33.0
# 3 DD        corz_br       24.8
# 4 DD        z_br         18.9
# 5 EDSS      corz_br       26.0
# 6 EDSS      z_br         19.8
# 7 PASAT     corz_br       37.1
# 8 PASAT     z_br         29.9
# 9 sex       corz_br       29.9
# 10 sex      z_br         22.7
# 11 type2    corz_br       22.1
# 12 type2    z_br         17.1

## Empirical lesion counts
lesion_counts <- colSums(lesions)
lesion_counts[lesion_counts == 0] <- NA
nifti_counts <- nifti(img = array(lesion_counts, dim(white_matter)))
lumin <- c(45, 100)
cols_counts <- heat_hcl(n = max(lesion_counts, na.rm = TRUE),
  h = c(265, 200),
  c = c(80, 50),
  l = lumin,
  power = c(0.7, 2))
overlay2.nifti(white_matter, y = nifti_counts, z = 32,
  plot.type = "single", plane = "sagittal",
  col.y = cols_counts, title = "lesion counts per voxel",
  col.main = "white")
```

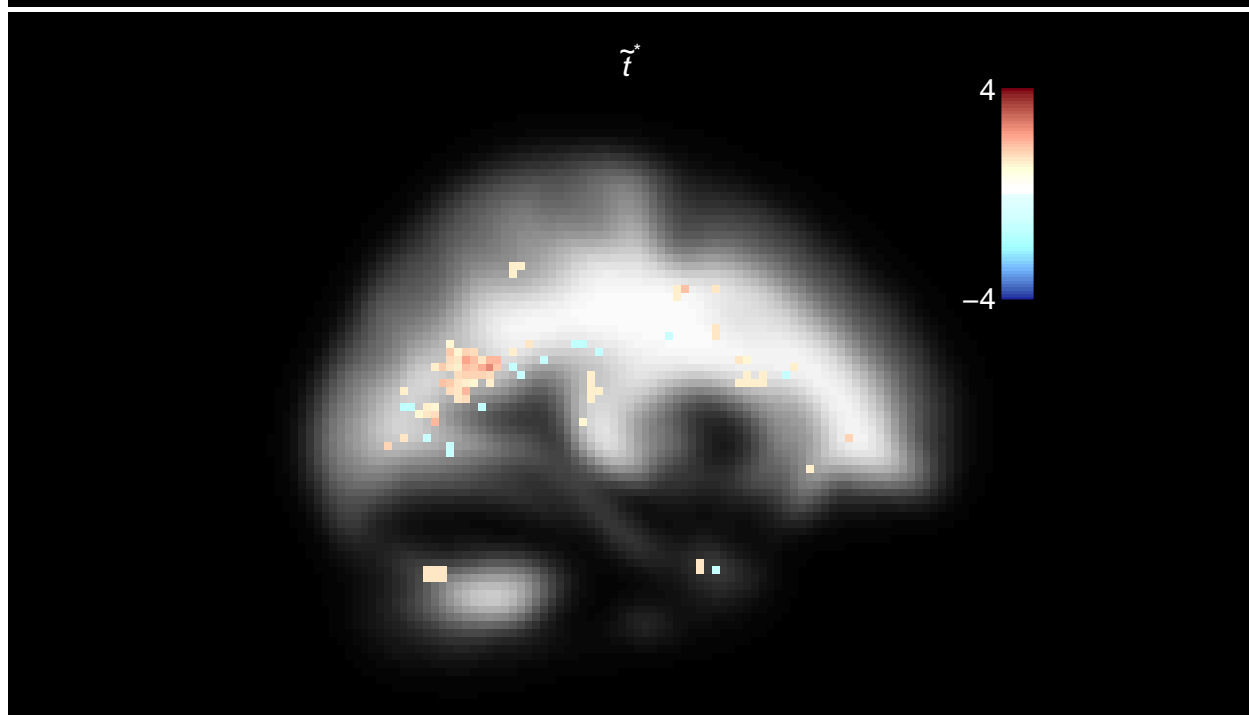
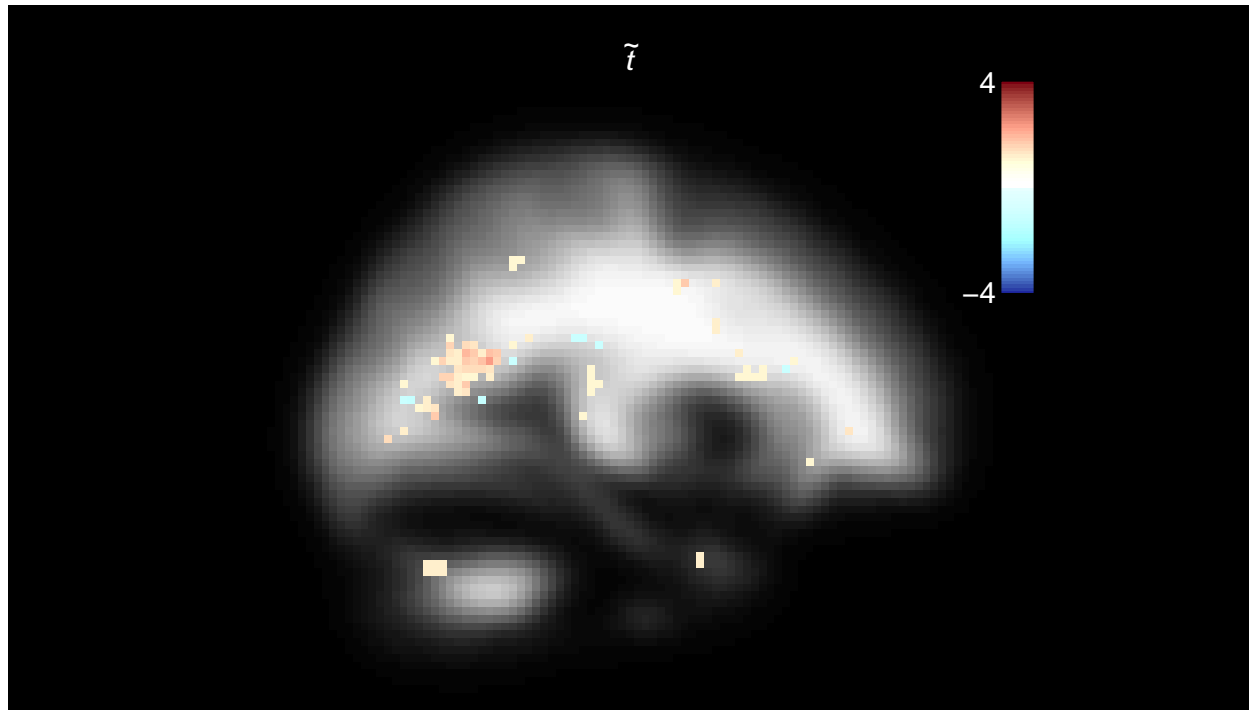



```
## Significance maps
param <- "DD"
low <- 1
upp <- 4
lumin <- c(25, 120)
cols <- c(heat_hcl(n = 32,
  h = c(265, 200),
  c = c(80, 50),
  l = lumin,
  power = c(0.7, 2)),
  rev(heat_hcl(n = 32,
    h = c(10, 40),
    c = c(80, 50),
    l = lumin,
    power = c(0.4, 1.3))))))
for (stat in c("z_br", "corz_br")) {
  zz <- (fits_mat %>% filter(statistic == stat & parameter == param))
  zz <- zz$value[array_indices]
  ## Threshold as in Ge et al (2014, AOAS)
  low_ind <- abs(zz) < low
  low_ind[is.na(low_ind)] <- FALSE
  zz[low_ind] <- NA
  upp_ind <- abs(zz) >= upp
  upp_ind[is.na(upp_ind)] <- FALSE
  zz[upp_ind] <- sign(zz[upp_ind]) * upp
  nifti_z <- nifti(img = array(zz, dim(white_matter)))
  nifti_z[1,1,1] <- -upp
  nifti_z[1,1,2] <- upp
  main <- switch(stat,
    z_br = expression(tilde(italic(t))),
```

```

    corz_br = expression(tilde(italic(t))~'*)
    overlay2.nifti(white_matter, y = nifti_z, z = 32, plot.type = "single",
    plane = "sagittal", col.y = cols, title = main,
    col.main = "white")
}

```



```

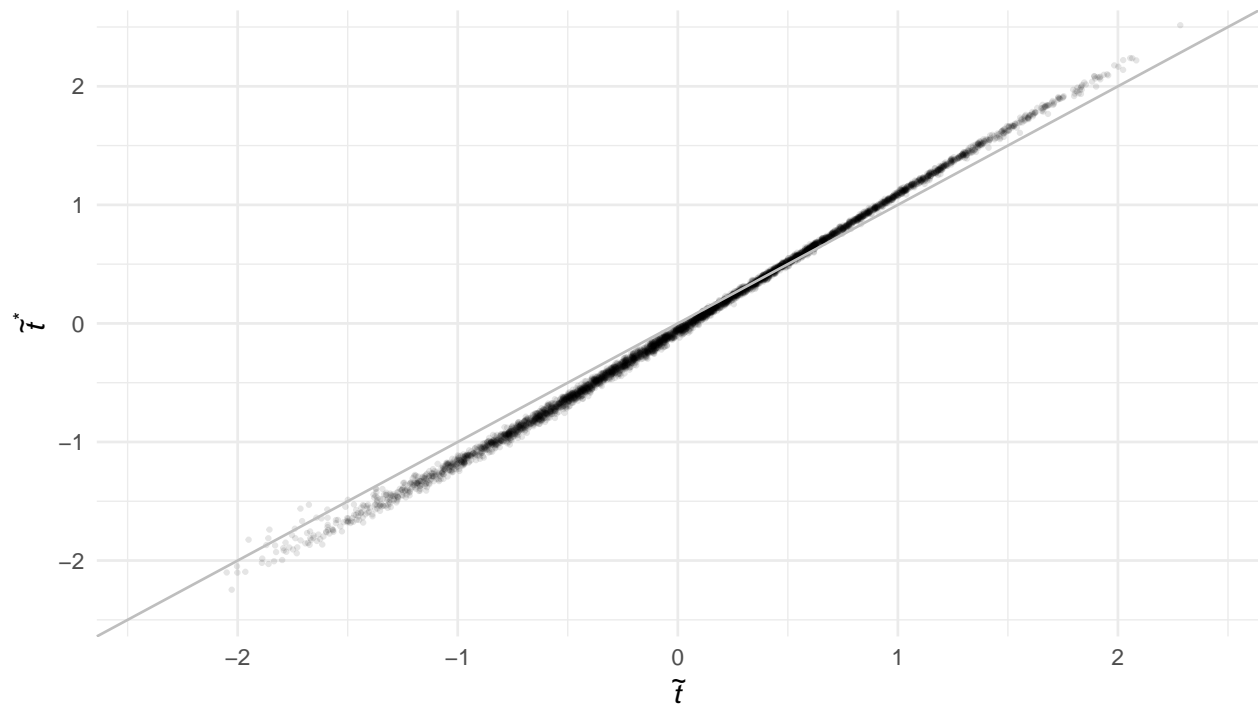
### Plot z_br vs corz_br per parameter
v1 <- fits_mat %>%

```

```

filter(statistic == "z_br", parameter == param) %>%
  select(z_br_value = value, voxel, parameter)
v2 <- fits_mat %>%
  filter(statistic == "corz_br", parameter == param) %>%
  select(corz_br_value = value, voxel, parameter)
v <- join(v1, v2, by = c("voxel", "parameter"))
ggplot(v) +
  geom_point(aes(x = z_br_value, y = corz_br_value), alpha = 0.1, size = 0.5) +
  geom_abline(aes(intercept = 0, slope = 1), col = "grey") +
  coord_cartesian(xlim = c(-2.4, 2.4), ylim = c(-2.4, 2.4)) +
  theme_minimal() +
  labs(x = expression(tilde(italic(t))), y = expression(tilde(italic(t))~''))

```



References

- Agresti, Alan, and Brian Caffo. 2000. "Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures." *The American Statistician* 54 (4). Taylor & Francis: 280–88. <https://doi.org/10.1080/00031305.2000.10474560>.
- Agresti, Alan, and Brent A. Coull. 1998. "Approximate Is Better Than Exact for Interval Estimation of Binomial Proportions." *The American Statistician* 52 (2). Taylor & Francis: 119–26. <https://doi.org/10.1080/00031305.1998.10480550>.
- Di Caterina, Claudia, and Ioannis Kosmidis. 2017. "Location-Adjusted Wald Statistic for Scalar Parameters." *ArXiv E-Prints*. <https://arxiv.org/abs/1710.11217>.
- R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.