Package and Function Glossary

- Libraries:
  - cowplot: Add-on to ggplot
  - dplyr: Data wrangling (mutate, select, filter, summarise)
  - readr: Dependency of tidyverse (read.csv, tsv, etc.)
  - tidyverse: Includes readr, dplyr, ggplot among others
  - repr: Create readable text and viewable images of data
  - infer: Used for statistical inference (specify, hypothesize, generate, calculate)
- Main functions:
  - **filter()**: Filters rows where conditions specified are true. Separate argument with ,
  - **select()**: Returns specific columns. Separate columns with, or add ranges as array [].
  - **rep_sample_n()**: Perform repeated sample of size n. can be used to make sample dists.
    - → size = size of each sample
    - → replace = can be TRUE or FALSE, false by default
    - → reps = number of samples
  - **pull()**: Pulls a single column from data frame as vector
  - **group_by()**: Converts tibble into grouped tibble based on the column value specified.
  - **summarise()**: Creates new column with transformed input (ex. mean = mean(column))
  - **get_ci()**: Returns a tibble with lower and upper ci
    - → level = confidence level
  - **specify()**: Specify variable or relationship between variables of interest
    - → formula = response ∼ explanatory
    - → Alternatively, can set response = and explanatory = to variables of choice. Both are needed during hypothesis testing, only response is needed during point estimate
    - → success = level of response considered a "success" (used primarily in proportion analysis)
  - **hypothesize()**: Declares hypothesis based on variables in specify
    - → null = null hypothesis. "independence" is used to determine relationship between response and explanatory. "point" is used to make point estimates
    - → mu/med/sigma = true parameter, used with point null hypothesis when response is continuous
  - **generate()**: Generates simulated distribution. For CIs, this is a bootstrap distribution. For hypothesis testing, this is a null distribution
    - → reps = number of resamples to generate
    - → type = method of generating resamples. "bootstrap" used to get bootstrap, "permute" used to get null dist (randomly assigns an input to a new output in each replicate).
  - **calculate()**: Returns statistic specified with stat argument
    - → stat = type of stat, such as mean, median, sum, sd, prop, diff in means/medians/props
    - → order = vector specifying the order in which explanatory variables should be subtracted, ie. c(first, second)
- **Sample Plot Workflows:**
  - **Histogram using ggplot:**

```
plot <- ... %>%
    ggplot(aes(x = ...)) +
    geom_histogram(bins/binwidth =
         ...) +
    ggtitle("...") +
    xlab("...")
```

  - **Boxplot using ggplot:**

```
plot <- ... %>%
    ggplot(aes(x = ..., y = ...))
        +
    geom_boxplot() +
    ggtitle("...") +
    ylab("...") +
     geom_hline(yintercept = ...,
         color = "blue")
```

  - → obs.stat is the observed stat from the sample/pop
  - → direction can be left, right, or two-sided
  - → endpoints is a tibble with lower and upper ci
- **Sampling Workflows:**
  - **Sampling from Population/Bootstrapping from Sample:**

```
estimates <- ... %>%
    rep_sample_n(size = ..., reps
        = ..., replace = ...) %>%
    group_by(replicate) %>%
    summarise(... = ...)
```

  - → replace = true for bootstrap, false for sampling
  - **Sampling then Selecting Variable:**

```
estimates <- ... %>%
    rep_sample_n(size = ..., reps
        = ..., replace = ...) %>%
    ungroup() %>%
    select(...)
```

  - **Selecting from filtered data:**

```
dataset <- ... %>%
    filter(... = ...) %>%
    select(...)
```

  - **Get CI from sample w/ infer package:**

```
dataset <- ... %>%
    summarise(lower_ci = quantile(
        stat, ...), upper_ci =
        quantile(stat, ...))
```

Probability and Events

- Sampling w/o replacement yields more precise parameter estimates, sampling w/ replacement guarantees independence.
- Sampling dist. shows all possible values of the sample mean for a given sample size and the likelihood of each value to appear
  - The shapes of the two are not necessarily the same/similar
  - Not all sample means are the same value as the pop mean
  - Sampling dist. is centred at the true population mean (may have difference variance)
  - Increasing sample size doesn't necessarily reduce variance of the sample dist. as variance of random sample is independent of size (it'll probably be closer to the population variance)
  - As the number of sampling reps increases, the distribution becomes more smooth (less missing values/gaps) - you can also use larger binwidths in the histogram
- Point estimates may be influenced by sampling bias (whether intentional or not)
- Standard error: used to quantify variation of point estimates in a sampling dist. (sd of sampling dist.). For bootstrapping, its sd of sds in each replicate, divided by the number of bootstraps.
- Estimator: rv whose dist. is the sampling dist. for a specific sample size and parameter
- Bootstrapping: useful for approximating a sampling dist. when we don't have access to the entire population
  - SD of bootstrap is a decent approximation of the sampling dist., differences between sample dist. and bootstrap are not influenced by sample size
  - It won't always be close to the SD or its corresponding sampling dist. as you might get an unlucky biased sample
  - Taking bootstraps larger than original results in an underestimated SE (artificially narrow), smaller bootstraps yields an overestimate
- Confidence interval: Indicates an X percent change that the true population parameter is between the upper and lower bounds

Simulation-Based Hypothesis Testing

- **Sample Plot Workflows:**
  - **Bootstrapping w/ Infer:**

```
samp_dist_mean <- ... %>%
     specify(response = ...) %>%
     generate(type = "bootstrap",
         reps = 10000) %>%
     calculate(stat = "mean")
```

  - **Point Hypothesis (mean/median/props) using Infer:**

```
null_model_infer <- ... %>%
    specify(response = ...) %>%
    hypothesise(null = "point", mu
         = mu_0) %>%
    generate(reps = ...) %>%
    calculate(stat = "mean/median/
        prop")
```

  - **Visualizing Hypothesis Test Results:**

```
null_model_vis_infer <- null_model
    _infer %>%
    visualize(..., bins/binwidth =
         ...) +
    shade_p_value(obs_stat = obs_
        test_stat, direction = "
        left") +
    xlab("...")
```

  - obs test stat is the test statistic (in these examples, it's the sample mean)
  - **Sampling from Null Distribution (diff in means/props):**

```
null_model <- ... %>%
    specify(formula = explanatory
        ~ response) %>%
    hypothesize(null = "
        independence") %>% #"
        independence" is used for
        diffs
    generate(reps = ..., type = "
        permute") %>%
    calculate(stat="diff␣in␣means/
        props", order = c("mu_1", "
        mu_2"))
```

- **Calculate p-value (difference):**

```
p_value <- ... %>%
    get_p_value(obs_stat = ...,
        direction = "both")
```

- **Sampling dist and Calc Z-score for each Replicate (sample mean):**

```
zscore_sample_means <- ... %>%
    rep_sample_n(reps = ..., size =
        ..., replace = FALSE) %>%
    group_by(replicate) %>%
    summarise(sample_mean = mean
        (...)) %>%
    mutate(z = sqrt(n) * (sample_
        mean - mu) / sigma )
```

  - In this case, mu and sigma are given to us from our initial sample
- **Histogram of Z-scores:**

```
sampling_dist_sample_mean_z <-
    zscore_sample_means %>%
    ggplot() +
    geom_histogram(aes(z, ..density
        ..), color = 'white',
        binwidth = ...) +  xlab("...
        ") +
    ggtitle("...")
```

- **All of the Above + Approximating Pop Statistics:**

```
n <- 5
sampling_dist_zscore_s <-
    ... %>%
    rep_sample_n(reps = ..., size =
        n, replace = FALSE) %>%
    group_by(replicate) %>%
    summarise(sample_mean = mean
        (...), sample_sd = sd(...))
        %>%
    mutate(z = sqrt(n) * (sample_
        mean - mu) / sample_sd) %>%
    ggplot() +  geom_histogram(aes(z
        , ..density..), color = '
        white', binwidth = ...) +
        xlab("...") + ggtitle("...")
```

- **One-Sample t-test + p-value (two-sided):**

```
## test stat
test_stat <-
    sqrt(nrow(...)) * (mean(...) -
        mu0) / sd(...)
p_value <- 2 * pt(test_stat, df =
    nrow(...) - 1, lower.tail =
    FALSE)
```

item **Recentering + p-value:**

```
samp_dist <- ... %>%
    specify(response = ...) %>%
    generate(type = "bootstrap",
        reps = ...) %>%
    calculate(stat = "mean")
null_model <- samp_dist %>%
    mutate(stat = stat - (mu - mu0))
         %>% #sample_mean - null
p_value <- mean(null_model$stat > mu
    )
```

- **Above using t.test():**

```
... <- tidy(
    t.test(..., mu = mu0)
    )
```

  - We need a vector/column of data points and mu = mu0

- **One-sample z-test (proportion):**

```
## This is 1-sided greater
phat <- mean(... == "...")
p0 <- 0.5
test_stat <-
    (phat - p0) / sqrt(p0 * (1-p0)/
        nrow(...))
p_value <-
    pnorm(test_stat, lower.tail =
        FALSE)
```

- **Above using prop.test():**

```
\item \textbf{One-sample z-test (
    proportion):}
        \begin{lstlisting}[language
            = R]
answer3.2.6 <-
    tidy(prop.test(x = sum(... == "
        ..."),
        n = nrow(...),
        p = 0.5,
        alternative = "...",
        conf.level = ...,
        correct=FALSE))
tidy(
    prop.test(
    x = # the number of successes,
    n = # the number of trials,
    p = # p0 (i.e., the value of p
        under H0),
    alternative = # alternative
        hypothesis: "less", "greater
        ", "two.sided"
    conf.level = # the desired
        confidence level,
    correct = FALSE))
```

- **Two-sample t-test (difference of means):**

```
## This is two-sided
... <- ... %>%
    filter(...) %>% #if cleaning
        needed
    group_by(...) %>%
    summarise(sample_mean = mean
        (...), sample_var = var(...)
        , n = n())
test_stat <- (...$sample_mean[2] -
    ...$sample_mean[1]) /
    sqrt(...$sample_var[2]/...$n[2]
        + ...$sample_var[1]/...$n
        [1])
p_value <- 2 * pt(test_stat, df =
    ..., lower.tail=FALSE)
```

- **Above using t.test():**

```
... <- tidy(
    t.test(x = ... %>% filter(... ==
         "...") %>% pull(...),
    y = ... %>% filter(... == "...")
         %>% pull(...),
    alternative = "two.sided")
    )
```

- **two-sample z-test (diff in props):**

```
qnts <- ... %>%
    group_by(...) %>%
    count(...)  %>%
    mutate(phat = n/sum(n))
n1 <- qnts %>%
    filter(... == "...") %>%
    pull(n) %>%
    sum()
n2 <- qnts %>%
    filter(... == "...") %>%
    pull(n) %>%
    sum()
phat1 <- qnts %>%
    filter(... == "..." & ... == "
        ...") %>%
    pull(phat)
phat2 <- qnts %>%
    filter(... == "..." & ... == "
        ...") %>%
    pull(phat)
phat <- (n1 * phat1 + n2*phat2)/(n1
    + n2)
test_stat <- (phat1 - phat2) / (sqrt
    (phat*(1-phat)*(1/n1 + 1/n2)))
p_value <- 2 * pnorm(test_stat,
    lower.tail = FALSE)
```

- **Above using prop.test():**

```
tidy(prop.test(x = c(n1*phat1, n2*
    phat2),
    n = c(n1, n2),
    correct = FALSE))
```

- **Paired t-test (means of two diff pops):**

```
... <- ... %>%
    mutate(d = after_... - before_
        ...)
test_stat <-
    sqrt(nrow(...))*mean(...$d)/sd
        (...$d)
p_value <- pt(test_stat, nrow(...) -
     1, lower.tail = FALSE)
```

- **Above using t.test():**

```
... <-
    tidy(t.test(x = ...$after_...,
        y = ...$before_...,
        paired = TRUE,
        alternative = 'greater'))
```

- **CI tibbles:**

```
,,,_ci <- tibble(
    lower_ci = qt(0.05, df = nrow
        (...) - 1) * ..._std_error +
         ..._x_bar,
    upper_ci = qt(0.95, df = nrow
        (...) - 1) * ..._std_error +
         ..._x_bar,
    )
..._clt_ci <-
    tibble(lower_ci = ..._x_bar +
        qnorm(...) * ..._std_error,
          upper_ci = ..._x_bar -
              qnorm(...) * ..._std_
              error)
```

- **Calculating type 1, type 2, and power:**

```
n <- ...
..._errors <- tibble(type_I_error =
    0.05,
    type_II_error = 1 - pnorm(qnorm
        (0.05, mu = mu0, sd = .../
        sqrt(n)), mu = mu1, .../sqrt
        (n)),
    power_of_test = pnorm(qnorm
        (0.05, mu0, .../sqrt(n)),
        mu1, .../sqrt(n)))
```

- **Finding proportion which reject null at each alpha level:**

```
n <- ...
... <- ... %>%
    mutate(test_statistic = sqrt(n)
        * (sample_mean - pop_mean) /
         pop_sd) %>%
    mutate(reject_h0 = abs(test_
        statistic) > qnorm(1-alpha/
        2)) %>%
    group_by(population, alpha) %>%
    summarise(proportion_rejection =
         mean(reject_h0)
```

- **ANOVA using aov():**

```
anova_results <- aov(formula =
    response ~ explanatory, data =
    ...) %>%
    tidy()

f_stat <- anova_results$statistic[1]

anova_pval <- anova_results$p.value
    [1]
```

- **Prediction interval (0.9), returns vector:**

```
... <- ... %>%
    pull(...) %>%
    quantile(c(0.05, 0.95)) %>%
    unname()
```

- **True Coverage Probability, returns vector:**

```
... <- ... %>%
    summarise(prop = mean(
        between( ..., ...[1],
            ...[2])
    )) %>%
    pull(prop)
```

- **All of the above + sampling**

```
sample_pi <- ... %>%
    rep_sample_n(100, replace =
        FALSE, reps = 1000) %>%
    group_by(replicate) %>%
    summarise(lower = quantile(...,
        0.05),
            upper = quantile(...,
                0.95))

..._vec <- ...$...

coverage <- sample_pi %>%
    group_by(replicate) %>%
    summarise(coverage = mean(
        between(..._vec, lower,
            upper)
        )) %>%
    pull(coverage)
```

- **Bonferroni Adjustment**

```
pval_bonf <- p.adjust(...$p_value,
    method = "bonferroni")
count_bonf <- sum(pval_bonf < 0.05)
```

- **BH Adjustment**

```
pval_bh <- p.adjust(...$p_value,
    method = "BH")
count_bh <- sum(pval_bh < 0.05)
```

- Pop and sample dists show how some individual data points are distributed, whereas sampling dists show how a statistic (aggregate of points) is distributed for a given sample size n
- Sampling Dist properties:
  - Centered at true population parameter
  - Bell-shaped/normal
  - Becomes narrower and more bell-shaped as n increases
- Standard error is measure of the variability of point estimates in sampling distribution, whereas sd (and variance) are measures of spread
- Test statistic: statistic to use for the test
- Null hypothesis: status quo. If true, the test statistic falls in the sampling distribution under H_0
- Alternative hypothesis: conclusion we wish to make (provided there's evidence to support it)
- p-value: Probability of getting a value at least as extreme as the observed one (left/right bound or two-sided)
  - There is an X chance of observing a test stat at least as extreme as the observed value, provided the null is true
- Significance level: Denoted as α, predetermined value so we reject the null is p-value $\leq \alpha$.
- Law of Large Numbers: LLN states that the sample averages converges to population mean as sample size increases (makes sense as the sample begins to more closely resemble the population)
- Normal (μ, σ) properties:
  - Unimodal and bell shaped
  - Symmetric around mean μ
  - SD σ quantifies spread
  - Z-score is calculated as $Z = \frac{(x-\mu)^2}{\sigma^2}$
  - Z-score means a value is Z-score SDs higher/lower than the mean
- CLT: for large sample size n, the sampling dist of the mean or proportion converges to the normal distribution REGARDLESS OF POPULATION
  - $\bar{X} \sim N(\mu, \frac{\sigma}{\sqrt{n}})$ or $\hat{p} \sim N(p, \sqrt{\frac{p(1-p)}{n}})$
- CLT Assumptions:
  - Items in sample are IID
  - Sample size is less than 0.1 of pop
  - Sample must be large enough np and $n(1-p)$ must be greater or equal to 10 for proportions, for means usually 30 is ok
- For means, if CLT is satisfied, we have $\bar{X} \sim N(\mu, \frac{\sigma}{\sqrt{n}})$. You can find this for 0.95 confidence using qnorm(0.975, $\bar{X}$, $\frac{s}{\sqrt{n}}$)
  - $CI(\mu, \alpha) = \bar{x} \pm z_{1-\frac{\alpha}{2}} * * \frac{s}{\sqrt{n}}$
- For means, if CLT is satisfied, we have $\hat{p} \sim N(p, \sqrt{\frac{p(1-p)}{n}})$
  - $CI(\hat{p}, \alpha) = \hat{p} \pm z_{1-\frac{\alpha}{2}} * * \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$
- For a difference in means, $\bar{X} - \bar{Y} \sim N(\mu_1 - \mu_2), \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$
  - $CI(\hat{p}, \alpha) = \hat{p} \pm z_{1-\frac{\alpha}{2}} * * \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$
- For a difference in proportions, $\hat{p}_1 - \hat{p}_2 \sim N(p_1 - p_2), \sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}$
  - $CI(\hat{p}, \alpha) = \hat{p} \pm z_{1-\frac{\alpha}{2}} * * \sqrt{\frac{\hat{p_1}(1-\hat{p_1})}{n_1} + \frac{\hat{p_2}(1-\hat{p_2})}{n_2}}$
- Standardizing a variable Z converts its distribution to standard normal
  - Apply the following transformation: $Z = \frac{\bar{X}-\mu}{\frac{\sigma}{\sqrt{n}}}$
  - For a proportion: $Z = \frac{\hat{p}-p_0}{\sqrt{\frac{p_0*(1-p_0)}{n}}}$
  - Use pnorm on the standardized Z value to get a Z value
- In the event we don't have the population sd, we can use s (sample sd). This adds extra variation.
  - $t = \frac{\bar{x}-\mu_0}{s/\sqrt{n}} \sim t_{n-1}$
  - The tails of the t-dist are heavier to account for the additional variation (for lower degrees of freedom)
  - T-dist is centered about 0 with a dof parameter. In 1-sample, dof = n - 1
  - The higher dof, the closer the t-dist is to N(0, 1)
- Steps for 1 proportion z-test:
  - Formulate null and alternative hypothesis (ex. $H_0 : p = 0.5$ vs $H_1 : p_1 \neq 0.5$)
  - Define test statistic (p)
  - Calculate null model $\hat{p} = N(p_0, \sqrt{\frac{p_0*(1-p_0)}{n}})$
  - Take sample
  - Calculate the observed test stat $\hat{p}$
  - Contrast observed test stat with null model by calculating p-value
- Two proportion z-test:
  - Formulate null and alternative hypothesis (ex. $H_0 : p_1 = p_2$ vs $H_1 : p_1 \neq p_2$)
  - Define test statistic $\hat{p_1} - \hat{p_2}$
  - Calculate null model $\hat{p} = N(0, \sqrt{p*(1-p)(\frac{1}{n_1} + \frac{1}{n_2})})$
  - Take sample x 2 and find $\hat{p}$
  - Calculate the observed test stat $\hat{p}_1 - \hat{p}_2$
  - Contrast observed test stat with null model by calculating p-value (2 * pnorm($\hat{p}_1 - \hat{p}_2$, 0, $\sqrt{p*(1-p)(\frac{1}{n_1} + \frac{1}{n_2})})$)
- Error in hypothesis testing faq:
  - Changing significance level does not affect effect size or overlap between SD and null (error). Location of border separating the null and alternative will change.
  - Increasing sample size reduces chance of type 2 error (power of test increases). The size of the overlap between null and alternative decreases
  - Lowering significance reduces chance of type 1 error, but increases type 2
  - By only reporting p-value, you miss info on effect size and error associated with the statistic
  - Increasing sample size increases power as it narrows the sampling dist, thus reducing overlap error between sampling dist and null
- Remember, var of a prop is $\frac{p(1-p)}{n}$
- ANOVA is used to compared between multiple groups, null is $\mu_1 = \mu_2 = ... = \mu_n$ (one variable is categorical, the other continuous)
  - ANOVA performs better if variances WITHIN groups are the same
  - If p value is low enough, conclusion is that at least 1 group has a mean that is significantly different from others (presence of difference vs quantifying that difference)
- Prediction interval: Interval when making inference on a new observation (not estimation from previous observations)
  - We take upper and lower quantiles of the population distribution.

.

- As sample size increases:
  - CIs get narrower as we zero in on true value
  - Prediction intervals remain roughly the same - we are getting better at estimating true quantiles making up the bounds of the PI
- True coverage probability is proportion of the pop that truly falls in the interval. If it's less than the specified probability:
  - More individuals will be outside the interval than we think
  - To capture the TCP, we have to shrink the PI
- The Bonferroni adjustment limits probability of seeing at least one Type 1 error to the specified significance level α to α/k (normally, probability of false positive is α)
  - This mitigates the difference between different types
- BH Adjustment controls the false discovery rate (rate significant features are actually null)